Université de Caen Basse-Normandie

École doctorale SIMEM

Thèse de doctorat

présentée et soutenue le : *21 novembre 2014*

par

Willy Ugarte Rojas

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

Spécialité : Informatique et applications

(Arrêté du 7 août 2006)

Extraction de motifs sous contraintes souples

Jury

Jean-François Boulicaut	Professeur	INSA de Lyon	(Rapporteur)
Lakhdar Sais	Professeur	Université d'Artois	(Rapporteur)
Christel Vrain	Professeur	Université d'Orléans	(Examinatrice)
Chedy Raïssi	Chargé de recherche	INRIA Nancy Grand-Est	(Examinateur)
Patrice Boizumault	Professeur	Université de Caen Basse-Normandie	(Directeur)
Bruno Crémilleux	Professeur	Université de Caen Basse-Normandie	(Directeur)
Samir Loudni	Maître de Conférences	Université de Caen Basse-Normandie	(Encadrant)

Remerciements

Tout d'abord, mes remerciements s'adressent aux personnes qui m'ont proposé le sujet de thèse et qui m'ont encadré tout au long de ces années d'étude : Patrice Boizumault, Bruno Crémilleux et Samir Loudni.

À travers de nos discussions, ils m'ont apporté une compréhension plus approfondie des divers aspects du sujet. Un grand merci à eux de m'avoir guidé, encouragé et conseillé. Je salue aussi l'ouverture d'esprit de mes directeurs de thèse qui ont su me laisser une large marge de liberté pour mener à bien ce travail de recherche.

Je suis très reconnaissant à Jean-François Boulicaut et Lakhdar Sais d'avoir accepté d'être rapporteurs de cette thèse, pour le temps précieux qu'ils ont consacré à cette tâche. Les commentaires et les questions, tant sur la forme du mémoire que sur son fond, ont contribué à améliorer de manière significative le document.

Je remercie aussi Christel Vrain et Chedy Raïssi d'avoir accepté de faire partie du jury, ses commentaires, remarques, questions et suggestions ont été une aide précieuse pour améliorer la compréhension de l'ensemble des travaux ainsi que la rédaction de la mémoire.

Je tiens à remercier mes amis de la faculté d'Ingénierie Informatique à Cusco qui ont toujours été présents malgré la distance, les amis et les professeurs de l'école Saint Francis d'Assise à Cusco et les amis que j'ai connu pendant toute cette expérience, spécialement à Francisco et à Maryshely qui m'ont accueilli plusieurs fois et m'ont fait sentir comme chez moi, faisant l'éloignement plus gérable.

Je souhaite aussi remercier tout le personnel du GREYC et du département informatique pour leur disponibilité et leur gentillesse.

Finalement, le plus important je remercie tous les membres de ma famille, en special : ma mère Judith, mon père Guillermo et mon frère Héctor qui m'ont toujours soutenu, malgré l'éloignement et qui m'ont donné toujours le meilleur d'eux, et en faisant l'effort d'être là le jour de ma soutenance.

Table des matières

Liste des tableaux

Table des figures

Liste des Algorithmes

Résumé

L'objectif de cette thèse est d'introduire de la souplesse dans le processus d'extraction de motifs en fouille de données. En utilisant la programmation par contraintes, nous avons pu apporter quatre principales contributions :

(1) La proposition d'un cadre général permettant de mettre en œuvre les contraintes souples de seuil dans un extracteur de motifs.

(2) L'introduction de la souplesse dans les skypatterns (motifs Pareto-optimaux par rapport à un ensemble de mesures) et la proposition d'une méthode générique permettant aussi bien l'extraction des skypatterns (durs) que des skypatterns souples.

(3) L'introduction du cube de skypatterns et la proposition de deux méthodes permettant sa construction : l'une, ascendante, repose principalement sur des règles de dérivation ; l'autre, utilise une approximation de l'ensemble des skypatterns du cube, rendue possible grâce aux skypatterns souples.

(4) L'introduction de la notion de motif optimal permettant de modéliser de nombreux problèmes d'extraction de motifs : skypatterns, top-k, motifs fermés, ... La déclarativité et la généricité de notre approche nous semblent ouvrir la voie à la définition et à la découverte de nouveaux ensembles de motifs.

Ces contributions ont été validées expérimentalement sur des domaines applicatifs réels tels que la découverte de toxicophores pour les deux premières contributions et la découverte de composants mutagènes pour la troisième.

Mots-clés: Extraction de motifs, CSP dynamiques, Contraintes (souples) de seuil, Skypatterns (souples), Cube de skypatterns, Motifs optimaux, Application en chemoinformatique

Abstract

The objective of this thesis is to introduce softness in pattern mining process in data mining. Using constraint programming, we were able to make four main contributions :

(1) A general framework for implementing soft threshold constraints in a pattern mining prototype.

(2) The introduction of softness in skypatterns (Pareto-optimal patterns with respect to a set of measures) and the proposal of a generic method for mining (hard) skypatterns as well as soft-skypatterns.

(3) The introduction of the skypattern cube and two methods for its construction : one bottom-up, mainly based on derivation rules ; the other uses an approximation of all skypatterns the cube, the method being feasible thanks to the soft-skypatterns.

(4) The introduction of the notion of optimal pattern for modeling many pattern extraction problems : skypatterns, top-k, closed patterns, ... The declarative and genericity side of our approach opens the way for the definition and discovery of new sets of patterns.

These contributions have been experimentally validated on real application domains such as the discovery of toxicophores for the first two contributions and the discovery of mutagenic components for third one.

Keywords: Pattern mining, Dynamic CSP, (Soft-) Threshold Constraints, (Soft-)Skypattern, Skypattern Cube, Optimal Patterns, Chemoinformatics application

Préambule

1 Contexte et positionnement

L'extraction de connaissances dans les bases de données - ou fouille de données - a pour objectif la découverte d'informations pertinentes, à partir de données brutes, généralement pour les présenter à un expert du domaine dont sont issues les données. Une étape centrale de ce processus est l'extraction de motifs, un motif exprimant une relation entre les données. L'utilisation de contraintes est très répandue pour indiquer le type de motifs à extraire afin de cibler le processus d'extraction suivant les centres d'intérêt de l'utilisateur.

La Programmation Par Contraintes (PPC) offre un cadre générique pour modéliser et résoudre les problèmes d'extraction de motifs en fouille de données. L'utilisation de la PPC pour la découverte de motifs en fouille de données est un domaine de recherche récent [De Raedt *et al.*, 2008]. Un des principaux atouts de la PPC est que l'utilisateur spécifie le problème d'extraction de motifs sous forme déclarative, par exemple un Problème de Satisfaction de Contraintes (CSP), et laisse le soin au solveur de calculer l'ensemble complet des solutions du problème.

Le premier article sur l'extraction de motifs à l'aide de la PPC a été publié en 2008 [De Raedt *et al.*, 2008] dans le cadre du projet CP4IM [1]. Les auteurs proposent un cadre unifié permettant de traiter une gamme de problèmes d'extraction de motifs locaux à l'aide de contraintes telles que la fermeture, la maximalité, les contraintes (anti-)monotones. Cette approche [Guns *et al.*, 2011], bien que fondatrice, est malgré tout limitée aux motifs locaux.

L'équipe CoDaG du GREYC a proposé deux approches utilisant la PPC pour l'extraction de motifs n-aires. Tout d'abord, [Khiari *et al.*, 2009, Khiari *et al.*, 2010a] ont proposé une approche générique, appelée *Hybride*, pour l'extraction de motifs sous contraintes n-aires, reposant sur l'utilisation conjointe d'un extracteur de motifs locaux et d'un solveur de CSP. [Khiari *et al.*, 2010b] propose une autre approche générique, appelée *Pure-CP*, pour l'extraction de motifs sous contraintes n-aires, reposant uniquement sur l'utilisation d'un solveur de CSP et étendant la modélisation proposée dans [De Raedt *et al.*, 2008]. Notons que [Guns *et al.*, 2013b] présente une méthode pour modéliser et résoudre des problèmes d'extraction portant sur les k-pattern sets, qui sont analogues aux motifs n-aires [Khiari *et al.*, 2010b].

[Métivier *et al.*, 2011, Métivier *et al.*, 2012] ont proposé un langage de contraintes permettant de formuler des requêtes de manière déclarative, et montrent comment l'utilisateur peut affiner ses requêtes pour parvenir à extraire les motifs ciblés. [Guns *et al.*, 2013a] ont défini, dans le cadre du projet ICON [2], le langage de contraintes MiningZinc qui, dans la lignée de MiniZinc [3] pour la PPC, permet de modéliser les problèmes d'extraction dans un langage de haut niveau et de générer des modélisations PPC utilisables par plusieurs solveurs.

1. Constraint Programming for Itemset Mining. https://dtai.cs.kuleuven.be/CP4IM/
2. Inductive Constraint Programming. http://www.icon-fet.eu/topics
3. http://www.minizinc.org/

2 Objectifs et motivations

Le point commun de l'ensemble des travaux, que nous venons de présenter, est de modéliser les problèmes d'extraction de motifs sous forme de CSP. Ainsi, un motif ne pourra être extrait que s'il vérifie toutes les contraintes.

Cependant, l'utilisateur aimerait pouvoir spécifier des préférences entre les motifs ou relâcher certaines contraintes. Par exemple, face à un ensemble de mesures, il est souvent très difficile de fixer les seuils des contraintes associées ; l'utilisateur aimerait pouvoir relâcher certains seuils afin de découvrir des motifs capturant un effet global sur l'ensemble des mesures mais ne vérifiant pas de façon stricte les contraintes sur un ou plusieurs seuils. De même, face à un problème sur-contraint (le CSP associé ne possède aucune solution), l'utilisateur aimerait pouvoir relaxer certaines contraintes, afin de pouvoir extraire des motifs, certes non "parfaits", mais qui peuvent, malgré tout, s'avérer pertinents.

Les travaux présentés dans ce mémoire ont pour objectif de s'attaquer aux deux constats suivants concernant l'extraction de motifs en fouille de données :

i) **La rigidité du cadre actuel** qui fait qu'une solution potentiellement intéressante n'est pas prise en compte dès qu'une contrainte et/ou une valeur seuil pour une contrainte sur une mesure n'est pas vérifiée. Par exemple, un biochimiste peut accepter un recouvrement entre deux molécules un peu supérieur au maximum demandé, si ce choix conduit à une bonne solution globale pour couvrir l'espace chimique.
   Il n'existe que de très rares travaux en fouille de données portant sur ce domaine. Pour les motifs locaux, [Bistarelli et Bonchi, 2007] ont proposé un premier cadre générique pour introduire une certaine souplesse dans les contraintes de seuil.

ii) **L'impossibilité pour l'utilisateur d'exprimer des préférences**, afin de pouvoir prendre en compte des priorités ou encore des incertitudes. Souvent, certains motifs s'avèrent plus intéressants que d'autres. Par exemple, un biochimiste peut préférer des molécules d'un certain type plutôt qu'un autre ; ou encore, lors de la construction d'un classiffieur pour prédire la toxicité, un biochimiste peut préférer privilégier certains attributs.
   Il existe très peu de travaux en fouille de données concernant les préférences exprimées par un utilisateur : citons les motifs les plus informatifs traduisant une relation de dominance locale entre motifs à l'aide d'une fonction de score [Pennerath et Napoli, 2009] ou encore, dans le cadre des bases de données, les skylines qui sont des points d'intérêt car non-dominés selon un ensemble de critères [Börzsönyi *et al.*, 2001].

3 Contributions

Grâce aux travaux existants en PPC sur la relaxation de contraintes et les préférences entre solutions, nous avons pu apporter quatre principales contributions :

1. La proposition d'un cadre général permettant de mettre en œuvre les contraintes souples de seuil dans un extracteur de motifs.
2. L'introduction de la souplesse dans les skypatterns (motifs Pareto-optimaux par rapport à un ensemble de mesures) et la proposition d'une méthode générique permettant aussi bien l'extraction des skypatterns (durs) que des skypatterns souples.
3. L'introduction du cube de skypatterns et la proposition de deux méthodes permettant sa construction : l'une, ascendante, repose principalement sur des règles de dérivation ; l'autre, utilise une approximation de l'ensemble des skypatterns du cube, rendue possible grâce aux skypatterns souples.

4. L'introduction de la notion de motif optimal permettant de modéliser de nombreux problèmes d'extraction de motifs : skypatterns, top-k, motifs fermés, ..., ainsi qu'une méthode PPC permettant d'extraire de tels motifs. La déclarativité et la généricité de notre approche nous semblent ouvrir la voie à la définition et à la découverte de nouveaux ensembles de motifs.

Ces contributions ont été validées expérimentalement :

a. En utilisant, tout d'abord, les jeux de données de l'*UCI* [4] afin de valider la faisabilité des méthodes proposées,
b. Puis, sur des domaines applicatifs réels tels que *la découverte de toxicophores* pour les deux premières contributions et *la découverte de composants mutagènes* pour la troisième contribution. Ces applications ont été réalisées grâce à la collaboration d'Alban Lepailleur du Centre d'Études et de Recherche sur le Médicament de Normandie (CERMN [5]).

Enfin, toutes nos contributions ont été mises en œuvre en utilisant le solveur de contraintes *Gecode* présenté à l'annexe A.1 (page 159).

Les quatre sections suivantes présentent plus en détail chacune de nos contributions.

3.1 Extraction de motifs sous contraintes souples de seuil

Dans la pratique, de nombreuses contraintes nécessitent des valeurs de seuil dont le choix est souvent arbitraire. Cette difficulté est encore plus aiguë lorsque plusieurs seuils sont nécessaires et sont combinés. En outre, les solutions (motifs) ratant de peu un seuil ne seront pas extraits même si elles s'avèrent pertinentes.

Tout d'abord, nous introduisons la notion de contrainte souple de seuil en définissant plusieurs sémantiques de violation. Puis, nous montrons comment de telles contraintes peuvent êtres mises en œuvre, dans un extracteur de motifs, en utilisant le modèle PPC de la relaxation disjonctive. Nous étendons alors cette approche à l'extraction des top-k motifs souples (i.e. les k motifs ayant les meilleures valeurs selon une mesure d'intérêt). Les expérimentations menées montrent l'apport des contraintes souples de seuil pour la découverte de fragments moléculaires toxicophores.

Publications : JFPC'12 [Ugarte *et al.*, 2012b], DS'12 [Ugarte *et al.*, 2012a] [6], JIIS'13 [Ugarte *et al.*, 2013b].

3.2 Extraction de skypatterns souples

Un skypattern est un motif qui n'est dominé par aucun autre motif (motif Pareto-optimal par rapport à un ensemble de mesures). Les skypatterns sont intéressants à double titre : ils ne nécessitent pas de seuil pour les contraintes portant sur les mesures ; de plus, la notion de dominance offre un intérêt global avec une sémantique facilement compréhensible par l'utilisateur. Mais, les skypatterns souffrent eux aussi de l'aspect dichotomique de la notion de dominance : des motifs proches de la frontière Pareto, mais qui ne sont pas des skypatterns, peuvent s'avérer être des motifs intéressants.

Tout d'abord, nous introduisons la notion de skypattern souple, en distinguant : les edge-skypatterns qui sont à la bordure de la zone de dominance, et les δ-skypatterns qui sont à une distance δ de cette bordure. Puis, nous proposons une méthode générique et efficace permettant d'extraire aussi bien les

4. `http://archive.ics.uci.edu/ml/`
5. `http://www.cermn.unicaen.fr/`
6. Carl Smith award du meilleur article jeune chercheur (`http://eric.univ-lyon2.fr/ds-2012/awards.html`).

skypatterns (durs) que les skypatterns souples. Cette méthode repose sur les CSP Dynamiques : l'idée majeure de notre approche est d'élargir pas à pas la zone de dominance par ajouts successifs de contraintes (de dominance). Ce processus s'arrête lorsque cette zone ne peut plus être agrandie. Les expérimentations menées montrent l'apport des skypatterns souples pour la découverte de toxicophores.

Publications : EGC'13 [Ugarte *et al.*, 2013a], LML'13 [Ugarte *et al.*, 2013c], CPAIOR'14 [Ugarte *et al.*, 2014d].

3.3 Extraction de cubes de skypatterns

Dans la pratique, les utilisateurs ne connaissent pas le rôle exact de chaque mesure et il est difficile de choisir à l'avance l'ensemble de mesures le plus approprié. Idéalement, les utilisateurs aimeraient garder toutes les mesures potentiellement utiles, et voir ce qui se passe sur un ensemble de skypatterns en supprimant ou en ajoutant une mesure pour en évaluer l'impact et ainsi converger vers un ensemble de skypatterns jugés pertinents. De manière analogue aux cubes de skylines dans les bases de données, les utilisateurs aimeraient pouvoir disposer de cubes de skypatterns.

Tout d'abord, nous introduisons le cube de skypatterns pour un ensemble de mesures M. Puis, nous proposons deux méthodes efficaces pour calculer un cube de skypatterns :

1. Une approche bottom-up, qui repose sur des règles de dérivation permettant de déterminer les skypatterns d'un nœud père à partir des skypatterns de ses nœuds fils, ceci sans effectuer aucun test de dominance. Les skypatterns non-dérivables sont calculés à la volée grâce aux CSP dynamiques. De plus, cette approche permet de construire, sans aucun effort supplémentaire, une représentation concise du cube fondée sur une relation d'équivalence entre ensembles de mesures (deux ensembles de mesures étant équivalents s'ils possèdent le même ensemble de skypatterns).
2. Une approche qui s'appuie sur la notion de skypattern souple. Cette méthode repose sur le fait que tout ensemble de skypatterns d'un nœud est inclus dans l'ensemble des edge-skypatterns de M. Le problème de construction du cube de skypatterns peut alors être facilement ramené au calcul d'un cube de skylines en $|M|$ dimensions, ce calcul étant effectué par un extracteur existant de cubes de skylines.

Les expérimentations menées sur une étude de cas en mutagénicité montrent l'intérêt de ces travaux.

Publications : ECAI'14 [Ugarte *et al.*, 2014b], ICTAI'14 [Ugarte *et al.*, 2014c].

3.4 Extraction de motifs optimaux

Tout d'abord, nous introduisons la notion de Motif Optimaux (MO), c'est-à-dire les meilleurs motifs selon une préférence définie par l'utilisateur, et nous montrons que les MO permettent de modéliser de nombreux problèmes d'extraction d'ensembles de motifs : libres, fermés, maximaux, skypatterns, top-k, motifs pics, miki, sous-groupes pertinents, etc. Puis, nous proposons une méthode générique pour extraire les MO. Cette méthode repose sur les CSP Dynamiques. L'idée majeure est la suivante : dès qu'une solution est trouvée, une contrainte est ajoutée dynamiquement au CSP courant afin de rechercher une solution de meilleure qualité (au sens de la préférence donnée par l'utilisateur), réduisant ainsi l'espace de recherche. Le processus s'arrête lorsqu'aucune meilleure solution ne peut être obtenue. En particulier, nous montrons que les MO sont caractérisés par une contrainte de base et un ensemble de contraintes ajoutées dynamiquement. Une étude expérimentale, menée sur plusieurs sortes de MO, illustre et discute la qualité des performances obtenues par rapport aux méthodes ad hoc.

Publications : RFIA'14 [Ugarte *et al.*, 2014a].

4 Organisation du mémoire

La première partie de ce mémoire est consacrée à un état de l'art des domaines centraux de notre étude que sont l'extraction de motifs sous contraintes et la PPC.

Le chapitre 1 introduit la problématique de l'extraction de motifs en fouille de données. Il résume les principales notions liées à l'extraction de motifs locaux et introduit les représentations condensées de motifs selon diverses mesures. Il présente ensuite la notion de préférence qui joue un rôle central pour nos contributions. Enfin, il dresse un bref panorama des principales approches d'extraction d'ensembles de motifs et positionne les résultats de nos travaux dans ce panorama.

Le chapitre 2 regroupe les notions de PPC nécessaires tout au long de ce mémoire. Il présente, tout d'abord, les notions de base relatives aux CSP. Puis, il s'intéresse successivement aux contraintes réifiées (utilisées pour modéliser les problèmes d'extraction de motifs), à la relaxation de contraintes en PPC (sur laquelle reposeront les contraintes souples de seuil), et enfin aux CSP dynamiques (qui permettront l'extraction des skypatterns, favoriseront la construction des cubes de skypatterns ainsi que l'extraction des motifs optimaux).

Le chapitre 3 s'intéresse à la modélisation, sous forme de CSP, des problèmes d'extraction de motifs sous contraintes et à l'encodage booléen du ou des motif(s) recherché(s). La modélisation d'une requête sous forme d'un CSP sera particulièrement utile pour modéliser et mettre en œuvre les contraintes souples de seuil. L'encodage booléen des motifs sera utilisé par chacune de nos contributions.

Le chapitre 4 est consacré à l'extraction des points skyline dans les bases de données multidimensionnelles et décrit différents algorithmes de calcul des skylines. Puis, il introduit la notion de cube de skylines ainsi que différentes méthodes pour le calculer.

La seconde partie décrit nos différentes contributions concernant l'apport de la souplesse pour l'extraction de motifs sous contraintes.

Le chapitre 5, introduit la notion de contrainte souple de seuil, et montre comment de telles contraintes peuvent êtres mises en œuvre, en utilisant le modèle PPC de la relaxation disjonctive. L'approche est alors étendue à l'extraction des top-k motifs souples. Les expérimentations menées montrent l'apport des contraintes souples de seuil pour la découverte de fragments moléculaires toxicophores [7].

Le chapitre 6 introduit la notion de skypattern souple et montre comment le problème de l'extraction de skypatterns (durs ou souples) peut être modélisé et résolu en utilisant les CSP dynamiques. Cette double contribution est validée expérimentalement sur des jeux de données de l'*UCI* et au travers d'une étude de cas consacrée à la recherche de toxicophores.

Le chapitre 7 introduit la notion de cube de skypatterns et propose deux méthodes pour construire un tel cube : tout d'abord, une méthode bottom-up fondée sur deux règles de dérivation, les skypatterns non-dérivables étant calculés à la volée ; puis, une seconde méthode utilisant les skypatterns souples, le problème de construction du cube de skypatterns pouvant être facilement ramené au calcul d'une cube de skylines. Ces deux méthodes sont validées et comparées expérimentalement sur des jeux de données de l'*UCI* et au travers d'une étude de cas consacrée à la découverte de composants mutagènes.

Le chapitre 8 introduit la notion de motifs optimaux, i.e. les meilleurs motifs selon une préférence définie par l'utilisateur, et montre que les MO englobent de nombreux problèmes d'extraction d'ensembles de motifs. Puis, une méthode générique pour extraire les MO est proposée. Une étude expérimentale, menée sur plusieurs sortes de MO et différents jeux de données de l'*UCI*, illustre et discute la qualité des performances obtenues par rapport aux méthodes ad hoc.

7. C'est dans ce chapitre qu'est présentée la problématique de la découverte de toxicophores (cf. la section 5.4.1).

Enfin, la conclusion dresse un bilan des travaux menés au cours de cette thèse. Puis, différentes perspectives sont présentées et discutées.

Première partie

État de l'art

"Si je ne peux pas l'imaginer, je ne peux pas le comprendre."

Albert Einstein

"Si vous torturez les données assez longtemps, elles vont finir par avouer."

Ronald Coase

Chapitre 1

Extraction de motifs

Sommaire

Ce chapitre introduit la problématique de l'extraction de motifs, à savoir la recherche de régularités (information) dans les jeux de données. La section 1.1 résume les principales notions liées à l'extraction de motifs locaux. La section 1.2 introduit les représentations condensées de motifs selon diverses contraintes. Nous présentons ensuite la notion de préférence (cf. la section 1.3) qui joue un rôle central pour nos contributions aux chapitres 6 et 8. La section 1.4 propose un bref panorama des principales approches d'extraction d'ensembles de motifs et plus généralement de motifs fondés sur des motifs locaux. Finalement, en section 1.5 nous mettons en exergue ce que nous considérons comme caractéristiques de ces approches et nous positionnons les résultats de nos travaux dans ce panorama.

1.1 Extraction de motifs locaux

Cette section introduit les termes et notions classiquement utilisées en extraction de motifs. Nous donnons des exemples usuels de contraintes et de mesures et nous explicitons comment la structuration de l'espace de recherche permet la mise en œuvre de techniques d'élagage pour l'extraction de motifs.

1.1.1 Cadre formel et définitions

Nous formalisons dans cette section les notions essentielles de *jeu de données*, *motif* et *langage*.

Comme vu en introduction, les données traitées dans ce mémoire sont des données transactionnelles. Une *transaction* est un objet d'étude des données décrite par des attributs booléens appelés *items*. Les données transactionnelles sont représentées sous la forme d'un tableau d'incidence ou d'existence à double entrée de dimension $m \times n$ croisant un ensemble $\mathcal{T} = \{t_l \mid 1 \leq l \leq m\}$ de m transactions avec un ensemble $\mathcal{I} = \{i_k \mid 1 \leq k \leq n\}$ de n items. La définition 1.1 précise la notion de *jeu de données*.

Définition 1.1. Jeu de données
Un jeu de données formel $\boldsymbol{r}$ est un triplet $(\mathcal{I}, \mathcal{R}, \mathcal{T})$ où :
- $\mathcal{I} = \{i_1, \ldots, i_n\}$ est l'ensemble des attributs (items),
- $\mathcal{T} = \{t_1, \ldots, t_m\}$ celui des objets (transactions) et
- $\mathcal{R}$ une relation binaire entre $\mathcal{I}$ et $\mathcal{T}$: $\mathcal{R}(i,t) = present$ (noté par 1) pour $i\mathcal{R}t$ et $\mathcal{R}(i,t) = absent$ (noté par 0) pour $\neg i\mathcal{R}t$.

Un jeu de données peut être vu comme une matrice binaire (cf. le tableau 1.1). Ce tableau représente un jeu de données $\boldsymbol{r}$ où 6 transactions étiquetées $t_1, \ldots, t_6$ sont décrites par 5 items $A, B, \ldots, E$. Dans cet exemple, chaque transaction appartient à une seule classe et est étiquetée par un item de classe (soit j_1, soit j_2). Autrement dit, $\boldsymbol{r}$ est partitionné en deux sous-ensembles, chaque sous-ensemble correspondant à une classe (on notera $\mathcal{T}_1$ l'ensemble des transactions de classe 1 et $\mathcal{T}_2$ celui de classe 2).

Exemple 1.1. Jeu de données

Classe	$\mathcal{T}$ \ $\mathcal{I}$	A	B	C	D	E
1	t_1	1	0	1	1	0
	t_2	0	1	1	0	1
	t_3	1	1	1	0	0
2	t_4	0	1	0	0	1
	t_5	1	1	1	0	1
	t_6	0	1	1	0	1

$\mathcal{T}$	Classe	Items
t_1	j_1	A C D
t_2	j_1	B C E
t_3	j_1	A B C
t_4	j_2	B E
t_5	j_2	A B C E
t_6	j_2	B C E

Table 1.1 – Deux représentations d'un jeu de données transactionnel $\boldsymbol{r}$.

L'objectif majeur de l'extraction de motifs est la découverte de relations utiles entre les objets de $\mathcal{T}$, chaque objet étant décrit par un sous-ensemble des items de $\mathcal{I}$. Ces relations, qui traduisent un comportement ou rendent compte d'un phénomène dans les données, reposent sur des *motifs* (aussi appelés *itemsets* dans le cas des données transactionnelles).

Définition 1.2. Motif
Un *motif ensembliste d'items* ou *itemset* est un sous ensemble de $\mathcal{I}$.

$\{A, C\}$, $\{B, C, E\}$, $\{C\}$ sont des exemples de motifs issus du jeu de données de la table 1.1. Pour alléger l'écriture, les motifs seront notés sous forme de chaînes plutôt que sous forme d'ensembles (i.e. AC au lieu de $\{A, C\}$). Nous précisons la notion de *taille* d'un motif :

Définition 1.3. taille
La *taille* d'un motif $\mathtt{x}$ est le nombre d'items que $\mathtt{x}$ contient (i.e. sa cardinalité) : $\mathtt{taille(x)} = |\mathtt{x}|$.

Un motif de taille k (i.e. constitué de k items) est appelé un k-motif (ou k-itemset). Sur le jeu de données de la table 1.1, il existe :
— 5 motifs de taille 1 : A, B, C, D et E.
— 10 motifs de taille 2 : AB, AC, AD, AE, BC, BD, BE, CD, CE et DE.
— 10 motifs de taille 3 : ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE et CDE.
— 5 motifs de taille 4 : $ABCD$, $ABCE$, $ABDE$, $ACDE$ et $BCDE$.
— 1 motif de taille 5 : $ABCDE$.

Une transaction $t \in \mathcal{T}$ supporte le motif $\mathtt{x}$ (ou $\mathtt{x}$ est présent dans t) si et seulement si $\mathtt{x} \subseteq t$ (ou $\forall i \in \mathtt{x}, R(i,t) = 1$).

Définition 1.4. Couverture
Étant donné un motif $\mathtt{x}$ et un ensemble de transactions $\mathcal{T}$, la couverture de $\mathtt{x}$ est l'ensemble des transactions qui le supportent : $\mathtt{couverture}(\mathtt{x}) = \{t \in \mathcal{T} \mid \mathtt{x} \subseteq t\}$

Le *langage* est une notion clé de l'extraction de motifs.

Définition 1.5. Langage
Le langage de motifs $\mathcal{L}_\mathcal{I}$ est l'espace de recherche des motifs construit sur l'alphabet $\mathcal{I}$ autrement dit c'est l'ensemble des parties de $\mathcal{I}$ (i.e. $2^\mathcal{I} \setminus \emptyset$).

Un jeu de données peut ainsi être défini comme un multi-ensemble de taille finie de motifs de $\mathcal{L}_\mathcal{I}$.

1.1.2 Contraintes locales et théorie

La notion de *contrainte* permet d'évaluer l'intérêt d'un motif selon des critères fixés par l'utilisateur. Une contrainte est un moyen à la fois de cibler les motifs recherchés par l'utilisateur et de réduire l'ensemble des motifs extraits afin de faciliter leur utilisation.

Définition 1.6. Contrainte
Une contrainte $\mathtt{c}$ est un prédicat défini sur un langage : $\mathtt{c} : \mathcal{L}_\mathcal{I} \rightarrow \{vrai, faux\}$ où $\mathtt{c}$ concerne un seul motif.

Nous dirons qu'une contrainte est *locale* ou *unaire* (et nous la notons avec un $\mathtt{c}$ minuscule) si elle porte sur un unique motif. Les motifs extraits avec des contraintes locales sont appelés *motifs locaux* [Hand, 2002, Morik *et al.*, 2005]. A contrario, nous verrons en section 1.4 des exemples de contraintes non unaires.

Il existe de nombreuses mesures utilisées dans les contraintes pour définir l'intérêt d'un motif [Geng et Hamilton, 2006]. Une des mesures les plus classiques est la *fréquence* [Agrawal et Srikant, 1994]. Cette dernière est le nombre d'occurrences d'un motif dans le jeu de données. Plus formellement :

Définition 1.7. Fréquence
La **fréquence**[8] est le cardinal de la couverture : $\mathtt{freq}(\mathtt{x}) = |\mathtt{couverture}(\mathtt{x})| = |\{t \in \mathcal{T} \mid \mathtt{x} \subseteq t\}|$

8. Le terme "fréquence" est ambigu dans la littérature : d'une part, il est utilisé pour désigner le nombre d'occurrences d'un motif (définition retenu dans ce mémoire), d'autre part, il est utilisé pour exprimer en pourcentage la fraction d'occurrences relativement au nombre total d'objets.

La contrainte de *motifs fréquents* est définie en sélectionnant les motifs dont la fréquence est supérieure à un seuil donné (cf. la définition 1.8).

Définition 1.8. Motif fréquent
Étant donné un seuil de fréquence minimale $minfr$ et un jeu de données $\boldsymbol{r}$, un motif $\mathtt{x}$ est fréquent dans $\boldsymbol{r}$ si et seulement si $\mathtt{freq}(\mathtt{x}) \geq minfr$

Dans l'exemple du tableau 1.1 pour $minfr = 3$ les motifs fréquents sont (nous indiquons entre $\langle . \rangle$ les valeurs de fréquence) : A $\langle 3 \rangle$, B $\langle 5 \rangle$, C $\langle 5 \rangle$, E $\langle 4 \rangle$, AC $\langle 3 \rangle$, BC $\langle 4 \rangle$, BE $\langle 4 \rangle$, CE $\langle 3 \rangle$ et BCE $\langle 3 \rangle$.

L'extraction de motifs sous contraintes consiste à extraire les motifs satisfaisant une contrainte $\mathtt{c}$ à partir d'un jeu de données $\boldsymbol{r}$. L'ensemble des motifs de $\mathcal{L}_\mathcal{I}$ vérifiant $\mathtt{c}$ dans $\boldsymbol{r}$ est appelé *théorie* [Mannila et Toivonen, 1997].

Définition 1.9. Théorie
Étant donné un langage $\mathcal{L}_\mathcal{I}$, un jeu de données $\boldsymbol{r}$ et une contrainte $\mathtt{c}$, la théorie $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})$ est l'ensemble des motifs de $\mathcal{L}_\mathcal{I}$ satisfaisant $\mathtt{c}$ dans $\boldsymbol{r}$: $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}) = \{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \mathtt{c}(\mathtt{x})\ est\ vrai\}$

Ce cadre formel permet d'unifier les différents types d'extraction quel que soit le contexte (jeu de données, langage, contrainte). Les motifs extraits suivant ce cadre sont *locaux* dans le sens que prouver qu'un motif $\mathtt{x}$ appartient à $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})$ ne requiert pas l'utilisation d'autres motifs.

Exemple 1.2.
L'extraction de motifs de taille minimale peut être définie avec une contrainte locale, soit ω un seuil de taille, ce problème consiste à extraire la théorie : $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, c) = \{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \underbrace{\mathtt{taille}(\mathtt{x}) \geq \omega}_{\mathtt{c}(\mathtt{x})}\}$

Afin de mieux spécifier les motifs recherchés, plusieurs contraintes peuvent être combinées formant ainsi une *requête* (par abus de langage, une contrainte est aussi appelée prédicat ou requête).

Définition 1.10. Requête
Une requête $\mathtt{q}$ est une conjonction de contraintes locales $\mathtt{c}_i$: $\mathtt{q}(\mathtt{x}) = \bigwedge \mathtt{c}_i(\mathtt{x})$

En poursuivant avec le jeu de données du tableau 1.1, l'ensemble des motifs qui satisfont la requête $\mathtt{q}(\mathtt{x}) = \mathtt{freq}(\mathtt{x}) \geq 3 \wedge \mathtt{taille}(\mathtt{x}) \geq 2$ est (les valeurs de fréquence et de taille sont indiquées entre $\langle . \rangle$) : AC $\langle 3, 2 \rangle$, BC $\langle 4, 2 \rangle$, BE $\langle 5, 2 \rangle$, CE $\langle 4, 2 \rangle$ et BCE $\langle 4, 3 \rangle$.

Les propriétés, notamment celles liées à (anti-)monotonicité des contraintes, ont largement été étudiées [Mannila et Toivonen, 1997]. La section 1.1.4 montre comment la propriété d'anti-monotonicité permet considérablement d'élaguer l'espace de recherche.

1.1.3 Exemples de mesures d'intérêt

Nous avons précédemment introduit les mesures de fréquence et de taille. Dans la pratique, il existe beaucoup de mesures d'intérêt portant sur des motifs locaux [Geng et Hamilton, 2006]. Une mesure d'intérêt vise à sélectionner et à classer les motifs en fonction de leur intérêt potentiel selon un certain point de vue. Plus formellement :

Définition 1.11. mesure
Étant donné un jeu de données $\boldsymbol{r}$, une **mesure** est une fonction qui associe une valeur à un motif de $\mathcal{L}_\mathcal{I}$: $\mathtt{m} : \mathcal{L}_\mathcal{I} \to \mathbb{R}$. Cette définition s'étend naturellement à un ensemble de motifs (cf. la section 1.4.2) : $\mathtt{m} : 2^{\mathcal{L}_\mathcal{I}} \to \mathbb{R}$ et aux motifs n-aires (cf. la section 1.4.1) : $\mathtt{m} : \mathcal{L}_\mathcal{I}^n \to \mathbb{R}$

Nous donnons maintenant quelques exemples de mesures utilisées dans la suite de ce mémoire. De nombreuses mesures sont construites à partir de la notion de fréquence d'un motif comme par exemple l'*aire* [Geerts *et al.*, 2004] :

Définition 1.12. Aire
L'**aire** d'un motif $\mathtt{x}$ est définie par : $\mathtt{aire}(\mathtt{x}) = \mathtt{freq}(\mathtt{x}) \times \mathtt{taille}(\mathtt{x})$

Utilisée dans une contrainte avec un seuil d'aire minimale, cette mesure permet de rechercher des motifs à la fois "suffisamment" fréquents et longs. Sur l'exemple du tableau 1.1, six motifs satisfont la contrainte $\mathtt{aire}(\mathtt{x}) \geq 6$. Il s'agit des motifs AC $\langle 6 \rangle$, BC $\langle 8 \rangle$, BE $\langle 8 \rangle$, CE $\langle 6 \rangle$, ABC $\langle 6 \rangle$ et BCE $\langle 9 \rangle$ (les valeurs entre $\langle . \rangle$ sont les valeurs de l'aire).

La recherche de contraste [Novak *et al.*, 2009] est particulièrement intéressante car elle permet d'obtenir des motifs dont la fréquence varie fortement d'un jeu de données à un autre. La mesure de *taux de croissance* (parfois aussi appelée *émergence*) est une mesure classique de recherche de contraste :

Définition 1.13. Taux de croissance ou émergence
Étant donné $\mathcal{T}_\mathtt{j}$ l'ensemble de transactions de classe $\mathtt{j}$ et $\mathtt{freq}_\mathtt{j}(\mathtt{x})$ la fréquence de $\mathtt{x}$ dans $\mathcal{T}_\mathtt{j}$, le **taux de croissance** ou **émergence** d'un motif $\mathtt{x}$ par rapport à la classe $\mathtt{j}$ est défini par :

$$\mathtt{émergence}_\mathtt{j}(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{freq}_\mathtt{j}(\mathtt{x}) = 0 \\ \infty & si \;\; \mathtt{freq}_\mathtt{j}(\mathtt{x}) > 0 \; et \; \mathtt{freq}(\mathtt{x}) = \mathtt{freq}_\mathtt{j}(\mathtt{x}) \\ \frac{|\mathcal{T} \setminus \mathcal{T}_\mathtt{j}| \times \mathtt{freq}_\mathtt{j}(\mathtt{x})}{|\mathcal{T}_\mathtt{j}| \times (\mathtt{freq}(\mathtt{x}) - \mathtt{freq}_\mathtt{j}(\mathtt{x}))} & sinon \end{cases}$$

Les motifs émergents introduits par [Dong et Li, 1999] découlent directement de cette mesure.

Définition 1.14. Motif émergent
Étant donné un seuil d'émergence ρ, un motif $\mathtt{x}$ est émergent pour $\mathcal{T}_\mathtt{j}$ si et seulement si $\mathtt{émergence}_\mathtt{j}(\mathtt{x}) \geq \rho$.

Les *Jumping Emerging Patterns* forment un cas particulier de motifs émergents. Leur spécificité est d'avoir un taux de croissance infini.

Définition 1.15. Jumping Emerging Pattern
Un motif $\mathtt{x}$ qui n'est présent que dans $\mathcal{T}_\mathtt{j}$ (i.e. $\mathtt{émergence}_\mathtt{j}(\mathtt{x}) = +\infty$) est appelé un *Jumping Emerging Pattern* (JEP).

Les motifs émergents et les JEPs caractérisent les classes de manière quantitative et qualitative. En faisant ressortir les distinctions entre classes, ils sont très utilisés dans la construction de classifieurs. Ils sont au cœur de nombreuses applications comme la classification de structures chimiques [Poezevara *et al.*, 2010]. Nous montrons l'apport de ces motifs en chemoinformatique à la section 5.6.2 (cf. page 79).

Sur notre exemple de la table 1.1, les six motifs A $\langle 2 \rangle$, AC $\langle 2 \rangle$, D $\langle \infty \rangle$, AD $\langle \infty \rangle$, CD $\langle \infty \rangle$ et ACD $\langle \infty \rangle$ sont émergents suivant la contrainte $\mathtt{émergence}_1(\mathtt{x}) \geq 2$ (les valeurs du taux de croissance sont indiquées entre $\langle . \rangle$). De plus, les motifs D, AD, CD et ACD sont des JEPs.

Dans la pratique, il est possible que des valeurs soient associées aux items (e.g. prix et/ou poids d'un produit, aromaticité d'un fragment, ...). Par exemple, la table 1.2 donne les valeurs d'un attribut associées aux items de notre exemple introduit à la table 1.1. Plusieurs attributs (et donc plusieurs valeurs) peuvent être associées à un item.

Il est alors intéressant de disposer de mesures comme les mesures d'agrégats portant sur ces valeurs. Soit val une fonction qui retourne la valeur d'un attribut à chaque item : pour un item i, $val(i)$ désigne

Item	A	B	C	D	E
Valeur	30	40	10	40	70

TABLE 1.2 – Valeurs associées aux items

la valeur d'un attribut associée à i. La table 1.3 indique quelques fonctions d'agrégats qui seront utilisées dans ce mémoire.

Mesure	Définition
`min (x)`	$\min_{i \in \mathtt{x}} \{val(i)\}$
`max (x)`	$\max_{i \in \mathtt{x}} \{val(i)\}$
`mean (x)`	$\frac{\mathtt{min(x)}+\mathtt{max(x)}}{2}$
`moyenne (x)`	$\frac{\sum_{i \in \mathtt{x}} \{val(i)\}}{\mathtt{\lvert x \rvert}}$

TABLE 1.3 – Agrégats appliqués aux valeurs associées aux items.

1.1.4 Espace de recherche

Le problème d'extraction de motifs est une tâche difficile du point de vue algorithmique de par la grande taille de l'espace de recherche. Pour les motifs ensemblistes, l'espace de recherche $\mathcal{L}_{\mathcal{I}}$ est un treillis (cf. la figure 1.1). Le motif le plus haut est l'ensemble vide, le niveau suivant (i.e. la deuxième ligne) contient les singletons, la troisième ligne les paires, etc. Le dernier niveau correspond au motif composé de l'ensemble de tous les items $\mathcal{I}$. Pour $|\mathcal{I}| = n$, la taille de l'espace de recherche est $2^n - 1$.

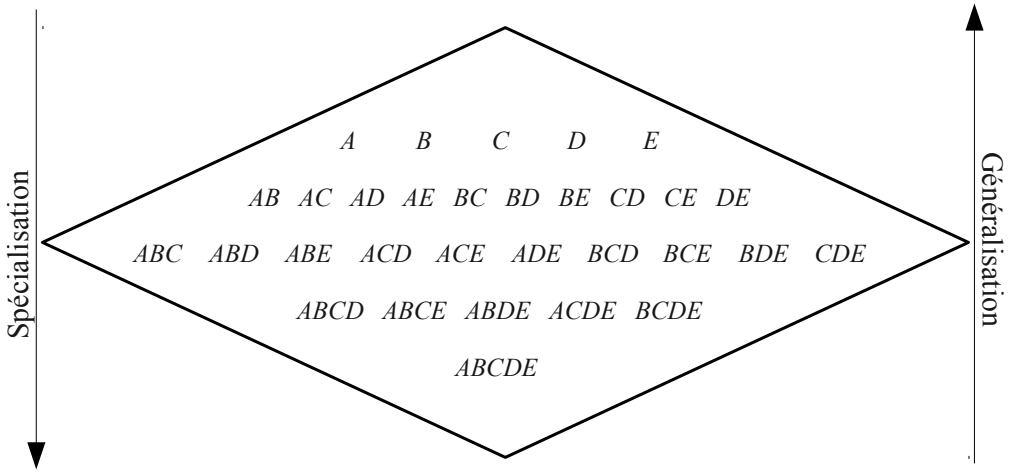


FIGURE 1.1 – Treillis représentant le langage $\mathcal{L}_{\mathcal{I}}$ avec $\mathcal{I} = \{A, B, C, D, E\}$.

Une relation de spécialisation [Mitchell, 1982] structure le langage $\mathcal{L}_\mathcal{I}$ et facilite la recherche des motifs.

Définition 1.16. Spécialisation - Généralisation
Une relation de spécialisation $\preceq$ est un ordre partiel défini sur les motifs de $\mathcal{L}_\mathcal{I}$. Un motif $\mathtt{x}$ est plus *spécifique* (respectivement plus *général*) qu'un motif $\mathtt{y}$ si $\mathtt{y} \preceq \mathtt{x}$ (respectivement $\mathtt{x} \preceq \mathtt{y}$).

Un exemple classique de *relation de spécialisation* est l'inclusion ensembliste $\subseteq$ entre deux motifs ensemblistes. Si un motif $\mathtt{x}$ est inclus dans un motif $\mathtt{y}$, on dit que $\mathtt{y}$ est plus spécifique que $\mathtt{x}$ ou que $\mathtt{x}$ est plus général que $\mathtt{y}$. Par exemple BE est inclus dans BCE et est donc plus général que BCE.

L'exploitation de la structure entre motifs dessinée par la relation de spécialisation, via les propriétés (d'anti-)monotonie, permet d'élaguer l'espace de recherche [Mannila et Toivonen, 1997]. Ces propriétés ont donné lieu à de nombreux travaux d'extraction de motifs sous contraintes et sont un élément majeur du succès de ce champ de recherche [Agrawal et Srikant, 1994, Ng *et al.*, 1998, Bonchi et Lucchese, 2007].

Définition 1.17. Contrainte anti-monotone
Étant donné un motif $\mathtt{x}$, une contrainte $\mathtt{c}(\mathtt{x})$ est *anti-monotone* par rapport à la relation de spécialisation $\preceq$ si et seulement si $\forall \mathtt{x}, \mathtt{y} \in \mathcal{L}_\mathcal{I},\ \mathtt{x} \preceq \mathtt{y} \implies (\mathtt{c}(\mathtt{y}) \implies \mathtt{c}(\mathtt{x}))$

Définition 1.18. Contrainte monotone
Étant donné un motif $\mathtt{x}$, une contrainte $\mathtt{c}(\mathtt{x})$ est *monotone* par rapport à la relation de spécialisation $\preceq$ si et seulement si $\forall \mathtt{x}, \mathtt{y} \in \mathcal{L}_\mathcal{I},\ \mathtt{x} \preceq \mathtt{y} \implies (\mathtt{c}(\mathtt{x}) \implies \mathtt{c}(\mathtt{y}))$

Pour simplifier, nous appellerons dans la suite *contrainte monotone* (respectivement *anti-monotone*) une contrainte monotone (respectivement anti-monotone) par rapport à la relation de spécialisation $\preceq$.

Exemple 1.3.
$\mathtt{c}(\mathtt{x}) \equiv \mathtt{freq}(\mathtt{x}) \geq minfr$ est un exemple de contrainte anti-monotone.
$\mathtt{c}(\mathtt{x}) \equiv \mathtt{taille}(\mathtt{x}) \geq \omega$ est un exemple de contrainte monotone.

L'(anti-)monotonie fournit une importante propriété d'élagage de l'espace de recherche :

Propriété 1 *Élagage fondé sur les contraintes (anti-)monotones.* Si un motif $\mathtt{x}$ ne satisfait pas la contrainte anti-monotone (respectivement monotone) $\mathtt{c}$, alors toutes ses spécialisations (respectivement généralisations) ne satisfont pas $\mathtt{c}$.

Autrement dit, si un motif $\mathtt{x}$ ne satisfait pas une contrainte anti-monotone $\mathtt{c}$, il est alors certain que toutes ses spécialisations ne vérifient pas $\mathtt{c}$ et l'exploration de l'espace de recherche peut être stoppée à partir de $\mathtt{x}$. De plus, si un sous-ensemble de $\mathtt{x}$ ne satisfait pas $\mathtt{c}$, $\mathtt{x}$ ne peut pas non plus satisfaire $\mathtt{c}$.

Les contraintes (anti-)monotones partitionnent l'espace de recherche en deux parties délimitées par des bordures [Mitchell, 1982, Mannila et Toivonen, 1997] : d'un côté, les motifs satisfaisant la contrainte et de l'autre, ceux ne la satisfaisant pas.

Définition 1.19. Bordure positive
La *bordure positive*, notée $\mathcal{B}d^+(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$, est l'ensemble des motifs $\mathtt{x}$ de $\mathcal{L}_\mathcal{I}$ tel que :

$$\mathtt{x} \in \mathcal{B}d^+(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})) \iff \mathtt{c}(\mathtt{x}) \wedge \forall \mathtt{y} \succeq \mathtt{x}, \neg \mathtt{c}(\mathtt{y})$$

La bordure positive est l'ensemble de motifs les plus spécifiques vérifiant $\mathtt{c}$, autrement dit $\mathcal{B}d^+(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$ rassemble les motifs maximaux de $\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})$.

De manière duale, nous définissons la bordure négative :

Définition 1.20. Bordure négative
La *bordure négative*, notée $\mathcal{B}d^-(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$, est l'ensemble des motifs $\mathtt{x}$ de $\mathcal{L}_\mathcal{I}$ tel que :

$$\mathtt{x} \in \mathcal{B}d^-(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})) \iff \neg\mathtt{c}(\mathtt{x}) \wedge \forall \mathtt{y} \preceq \mathtt{x}, \mathtt{c}(\mathtt{y})$$

La bordure négative est l'ensemble de motifs les plus généraux ne vérifiant pas $\mathtt{c}$, autrement dit $\mathcal{B}d^-(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$ rassemble les motifs minimaux de $\mathcal{L}_\mathcal{I} \setminus \mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})$.

La figure 1.2 indiquent les bordures positives et négatives pour la contrainte anti-monotone de fréquence $\mathtt{freq} \geq 3$, toujours à partir du jeu de données du tableau 1.1 (page 12).

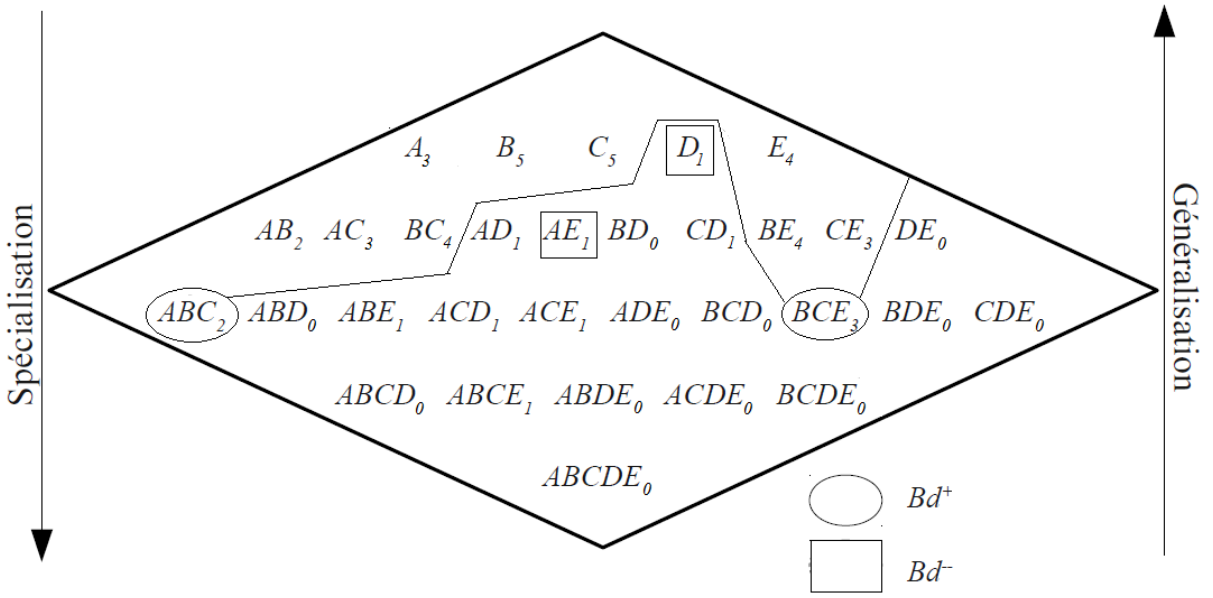


Figure 1.2 – $\mathcal{B}d^+(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$ et $\mathcal{B}d^-(\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c}))$ pour $\mathtt{c}(\mathtt{x}) = \mathtt{freq}(\mathtt{x}) \geq 2$

Cependant, de nombreuses contraintes utiles dans les problèmes d'extraction de motifs ne vérifient pas de propriétés de monotonie ou d'anti-monotonie, c'est par exemple le cas des motifs émergents (cf. la définition 1.14). En effet, le taux de croissance est un ratio de fréquences. Lors de la spécialisation d'un motif $\mathtt{x}$, les diminutions des fréquences de $\mathtt{x}$ sur chaque classe ne sont pas nécessairement similaires et le ratio peut augmenter ou diminuer. Ainsi, pour le jeu de données du tableau 1.1, avec un seuil d'émergence $\rho = 1.5$ et pour la classe 2, le motif B est émergent ($\mathtt{émergence}_2(B) = \frac{3\times3}{3\times2} = 1.5$), le motif BC ne l'est pas ($\mathtt{émergence}_2(BC) = \frac{3\times2}{3\times2} = 1$) alors que le motif BCE est émergent ($\mathtt{émergence}_2(BCE) = \frac{3\times2}{3\times1} = 2$). La contrainte d'émergence est ainsi plus difficile à mettre en œuvre.

Il existe différentes approches pour extraire des motifs sous cette contrainte, telles que la manipulation de bordures [Dong et Li, 1999] ou par décomposition des éléments composants la contrainte afin de se ramener à des sous-espaces où la propriété de monotonicité est vérifiée [Soulet et Crémilleux, 2005, Cerf, 2010].

1.1.5 Conclusion

Une limite bien connue de l'extraction de motifs locaux est le nombre de motifs qui peut être produit. Selon la requête, celui-ci peut être très grand (plusieurs millions, milliards, etc. on parle alors de "déluge de motifs") rendant ainsi impossible la tâche d'interprétation et d'exploitation des motifs par l'utilisateur. La suite de ce chapitre présente plusieurs notions (représentations condensées, préférences, ensembles de motifs) pour à la fois réduire l'ensemble de motifs obtenus et produire des motifs plus utiles à l'utilisateur. Ces notions seront reprises dans les différentes contributions de ce mémoire

1.2 Représentations condensées de motifs

Les *représentations condensées* de motifs satisfaisant une contrainte `c` [Mannila et Toivonen, 1996] ont été introduites pour augmenter la rapidité d'exécution des algorithmes d'extraction de motifs et réduire le nombre de motifs obtenus. Les algorithmes de production de représentations condensées de motifs reposent sur le principe des classes d'équivalence : l'idée centrale est de n'extraire qu'un nombre réduit de motifs à partir desquels il est possible de régénérer, si on le souhaite, tous les motifs vérifiant `c`. Il existe deux types de représentations condensées : les représentations exactes et celles dites approximatives. Avec une représentation exacte, il est possible de régénérer pour chaque motif les valeurs exactes des mesures associées à `c` (par exemple la fréquence dans le cas des motifs fréquents) alors que cette valeur est connue seulement avec une approximation contrôlée dans le cas des représentations condensées approximatives.

Définition 1.21. Représentation condensée
La représentation condensée d'un ensemble de motifs $\mathcal{E}$ selon une contrainte `c` est un ensemble de motifs $\mathcal{E}'$ tel que $\mathcal{E}' \subseteq \mathcal{E}$ et $\forall x \in \mathcal{E} \setminus \mathcal{E}'$, `x` peut être déduit (efficacement) à partir de $\mathcal{E}'$.

Pour les motifs ensemblistes et la contrainte de fréquence minimale, les deux représentations condensées les plus classiques sont celles fondées sur les *motifs fermés* [Pasquier *et al.*, 1999a] (définition 1.22), aussi appelés *motifs clos*, et les *motifs libres* (définition 1.23) [Boulicaut *et al.*, 2000].

Définition 1.22. Motif fermé
Un motif `x` est fermé si et seulement si toutes ses spécialisations strictes ont une fréquence strictement inférieure à celle de `x` : $\texttt{fermé}(\texttt{x}) \iff \forall \texttt{y} \supset \texttt{x},\ \texttt{freq}(\texttt{y}) < \texttt{freq}(\texttt{x})$

Définition 1.23. Motif libre
Un motif `x` est libre si et seulement si toutes ses généralisations strictes ont une fréquence strictement supérieure à celle de `x` : $\texttt{libre}(\texttt{x}) \iff \forall \texttt{y} \subset \texttt{x},\ \texttt{freq}(\texttt{y}) > \texttt{freq}(\texttt{x})$

Les motifs libres et fermés structurent le treillis des motifs en *classes d'équivalence*. Une classe d'équivalence est délimitée par un ou plusieurs motifs libres qui constituent ses éléments minimaux et par un seul motif fermé qui constitue son élément maximal (minimaux et maximal s'entendent au sens de la taille des motifs). Tous les motifs d'une même classe d'équivalence ont la même couverture. La figure 1.3 représente les classes d'équivalence ainsi que les motifs libres et fermés pour le treillis de la figure 1.1 (page 16) correspondant au jeu de données du tableau 1.1 (page 12) selon la contrainte $\texttt{freq}(\texttt{x}) \geq 2$. Cette figure illustre le fait que cet exemple contient 19 motifs fréquents qui sont résumés par 9 classes d'équivalence (et donc 9 motifs fermés). A partir de ces 9 motifs fermés et de leurs fréquences, l'ensemble de tous les motifs fréquents (suivant la contrainte $\texttt{freq}(\texttt{x}) \geq 2$) peut être régénéré : la fréquence d'un motif `x` est la même que celle du fermé de sa classe d'équivalence. Par exemple, la fréquence de D est celle de ACD, ainsi $\texttt{freq}(D) = \texttt{freq}(ACD) = 1$. Cet exemple contient 9 motifs libres à partir desquels il est aussi possible de régénérer la fréquence de chaque motif fréquent. Le fait que chaque classe d'équivalence a ici un seul motif libre est un cas particulier.

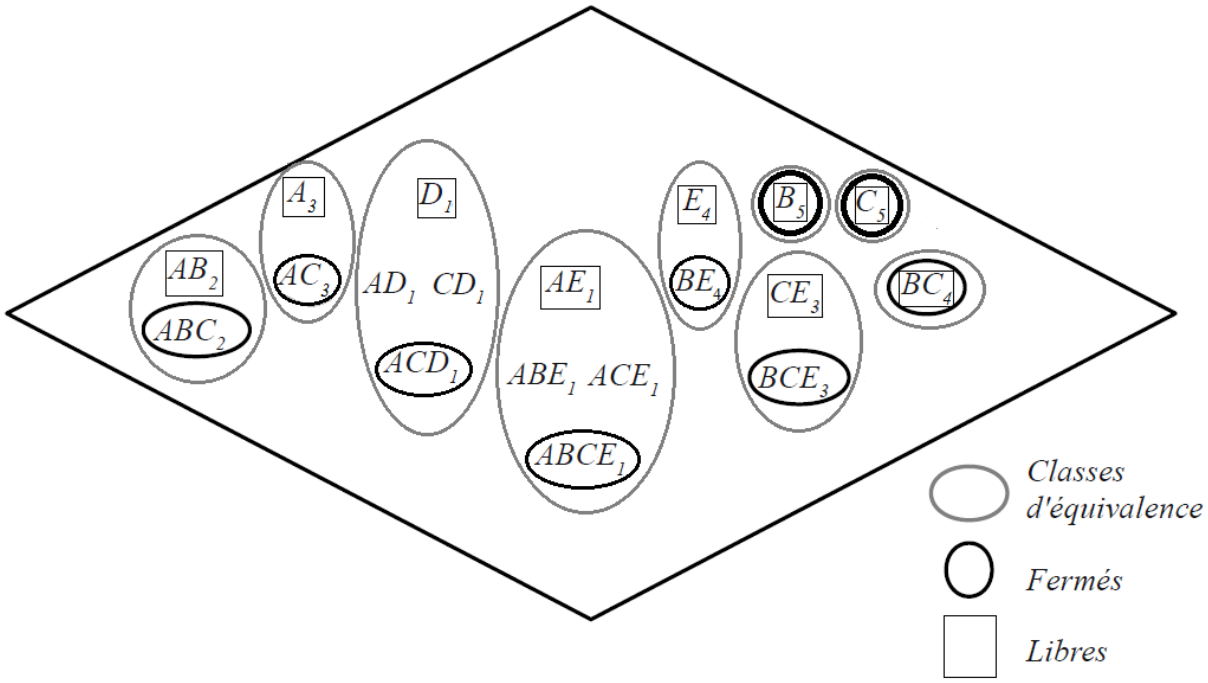


FIGURE 1.3 – Classes d'équivalence pour la contrainte `freq(x)` ≥ 1 pour le jeu de données du tableau 1.1.

Il existe de nombreuses propositions pour le calcul des motifs libres, fermés et leurs représentations condensées associées. Pour les motifs fermés, citons les algorithmes `A-CLOSE` [Pasquier *et al.*, 1999a] qui a été étendu en `CLOSE` [Pasquier *et al.*, 1999b], `CHARM` [Zaki et Hsiao, 2002] ou encore `LCM` [Uno *et al.*, 2004] qui est réputé comme l'un des algorithmes les plus efficaces pour cette tâche. Une caractéristique intéressante des motifs libres est qu'ils vérifient la propriété d'anti-monotonicité et peuvent ainsi être extraits en s'appuyant sur celle-ci [Boulicaut *et al.*, 2000]. Il existe plusieurs extensions de la notion de motifs libres tels que les motifs k-libres [Calders et Goethals, 2007] et les motifs δ-libres [Boulicaut *et al.*, 2003]. Ces derniers fournissent une représentation condensée approximative des motifs (celle-ci est plus concise qu'une représentation exacte, mais au prix d'une certaine approximation sur les valeurs de fréquences associées aux motifs). Une autre représentation approximative est celle des *motifs maximaux*, c'est-à-dire des motifs formant la bordure positive (cf. la définition 1.19) [Burdick *et al.*, 2005, Gouda et Zaki, 2005]. Cette représentation est approximative car elle ne permet pas de dériver la fréquence exacte de chaque motif fréquent, mais seulement une borne inférieure.

Les représentations condensées précédentes s'appuient sur la notion de fermeture de Galois et sont bien adaptées aux contraintes définies sur des mesures fondées sur la fréquence comme par exemple les mesures de contrastes [Soulet *et al.*, 2004, Li *et al.*, 2007]. D'autres opérateurs de fermeture peuvent être utilisées. [Soulet, 2006] introduit un opérateur de fermeture préfixée qui produit une représentation condensée sous forme d'intervalles disjoints. La représentation obtenue possède une propriété remarquable : chaque motif vérifiant la contrainte `c` qui définit la représentation condensée est présent dans un unique intervalle de la représentation condensée. L'algorithme `MICMAC` (Minimal constrained and Maximal constrained patterns) [Soulet et Crémilleux, 2008] définit un opérateur de fermeture adéquat aux mesures utilisées pour extraire les motifs libres et les motifs fermés selon ces mesures. L'opérateur de fermeture adéquat est automatiquement inféré à partir des mesures utilisées. Cette méthode permet d'extraire les motifs libres et fermés suivant un large ensemble de mesures (telles que la fréquence, la fréquence disjonctive, mais aussi des mesures comme *min*, *max*, *sum*, *count*, des exemples sont donnés en section 1.1.3) et non plus seulement suivant les mesures fondées sur la fréquence. L'algorithme fonctionne par niveaux et exploite la

propriété d'anti-monotonie liée aux motifs libres. Au chapitre 6, nous utilisons ce principe pour extraire les représentants des motifs skylines suivant des mesures.

1.3 Préférences

La découverte de motifs sous contraintes permet d'évaluer la pertinence et la qualité des motifs par rapport à de nombreuses contraintes et mesures proposées dans la littérature. Cependant, dans la pratique, l'utilisation de contraintes requiert souvent l'usage de seuils qu'il est difficile de fixer. Considérons l'exemple 1.4 :

Exemple 1.4.
Soit $\boldsymbol{r}$ une base de données d'un supermarché et supposons que le propriétaire du magasin souhaite trouver des achats assez fréquents, contenant au moins un produit cher et ayant un montant total donné. Cette requête peut être exprimée dans le cadre de l'extraction de motifs sous contraintes, mais en fixant des seuils (ici : $minfr$, $\psi_{prixunitaire}$ et $\psi_{prixtotal}$) :

— trouver des achats assez fréquents : $\texttt{freq}(\texttt{x}) \geq minfr$ (e.g. $minfr = 50$),
— contenant au moins un produit cher : $\max\limits_{i \in \texttt{x}} (i.prix) \geq \psi_{prixunitaire}$ (e.g. $\psi_{prixunitaire} = 100$),
— ayant un montant total donné : $\sum\limits_{i \in \texttt{x}} i.prix \geq \psi_{prixtotal}$ (e.g. $\psi_{prixtotal} = 500$),

Le succès - ou pas - de cette requête dépend fortement de la capacité de l'utilisateur à fixer ces seuils. Dans la pratique, l'utilisateur détermine souvent les seuils en tâtonnant et en lançant plusieurs extractions, ce qui est particulièrement fastidieux. De plus, il n'y a pas de garantie qu'un motif intéressant ne satisfasse pas un seuil : ce motif n'est alors malheureusement pas produit. Nous verrons au chapitre 5 (cf. page 59) que les contraintes souples de seuil sont un premier élément de réponse à ce problème.

Le fond du problème est que la spécification de contraintes, et particulièrement le choix des seuils, est une tâche qui va au-delà des capacités de l'utilisateur. Une alternative est de demander à l'utilisateur de simplement spécifier des *préférences*. Récrivons l'exemple 1.4 sous forme de préférences :

Exemple 1.5.
"plus le motif est fréquent, plus le prix d'un des produits du motif est élevé et plus le montant total des items du motif est élevé, meilleur est le motif"

Clairement, il est plus simple d'exprimer une requête sous forme de préférence que de fixer des seuils. Nous donnons maintenant une définition formelle de la notion de préférence :

Définition 1.24. Préférence
Une **préférence** $\rhd$ est une relation d'ordre strict [9] et partiel [10] sur $\mathcal{L}_{\mathcal{I}}$. Soient $\texttt{x}$ et $\texttt{y}$ deux motifs, $\texttt{x} \rhd \texttt{y}$ indique que $\texttt{x}$ est strictement préféré à $\texttt{y}$.

Exemple 1.6.
A partir de l'exemple 1.5 on a :

$$\texttt{x} \rhd \texttt{y} = \texttt{freq}(\texttt{x}) > \texttt{freq}(\texttt{y}) \wedge \max_{i \in \texttt{x}} \{i.prix\} > \max_{j \in \texttt{y}} \{j.prix\} \wedge \sum_{i \in \texttt{x}} i.prix > \sum_{j \in \texttt{y}} j.prix$$

Une préférence étant une relation d'ordre partiel, des motifs peuvent être incomparables (i.e. $\texttt{x} \not\rhd \texttt{y}$ et $\texttt{y} \not\rhd \texttt{x}$.)

9. Une relation binaire est une relation d'ordre strict quand elle est irréflexive et transitive.
10. Il existe deux motifs $\texttt{x}$ et $\texttt{y}$ ($\texttt{x} \neq \texttt{y}$) qui sont incomparables par rapport à $\rhd$ (i.e. $\texttt{x} \not\rhd \texttt{y}$ et $\texttt{y} \not\rhd \texttt{x}$).

Par exemple, si un client a acheté l'ensemble de produits $\mathtt{x}$ qui contient le produit le plus cher et un autre client a acheté l'ensemble de produits $\mathtt{y}$ tel que $\sum_{j \in \mathtt{y}} j.prix > \sum_{i \in \mathtt{x}} i.prix$, alors $\mathtt{x}$ et $\mathtt{y}$ sont incomparables (i.e. $\mathtt{x} \not\succcurlyeq \mathtt{y}$ et $\mathtt{y} \not\succcurlyeq \mathtt{x}$) car $\max_{i \in \mathtt{x}} \{i.prix\} > \max_{j \in \mathtt{y}} \{j.prix\}$ et $\sum_{j \in \mathtt{y}} j.prix > \sum_{i \in \mathtt{x}} i.prix$.

On notera que la notion de préférence (tout comme celle de contrainte) est ici liée à la notion de mesure (cf. la section 1.1.3) puisqu'une préférence signifie qu'une valeur d'une mesure pour $\mathtt{x}$ est supérieure (ou pas) à la valeur de cette mesure pour $\mathtt{y}$. Plus généralement, cette voie de recherche connaît un intérêt accru depuis quelques années. De nouvelles méthodes de fouille sont apparues, inspirées de l'analyse multi-critère, comme le calcul des motifs skylines qui s'apparente à la recherche de la frontière de Pareto [Soulet *et al.*, 2011]. Nous contribuons de plusieurs façons à cette vois de recherche. Au chapitre 6, nous montrons l'apport de la programmation par contraintes à la découverte de motifs skylines et nous proposons les motifs skylines souples. Le choix des mesures à utiliser peut être une limite de l'usage des préférences. Aussi, dans le cas des motifs skylines, nous proposons de conserver toutes les mesures et nous construisons le cube des motifs skylines qui permet de mieux comprendre leur rôle et ainsi de faire ressortir les ensembles de motifs les plus utiles. Enfin, la notion de préférence est au cœur de nos contributions sur les motifs optimaux (cf. le chapitre 8).

1.4 Motifs fondés sur des motifs locaux

Comme nous l'avons vu à la section 1.1, l'intérêt d'un motif local ne dépend pas des autres motifs produits. Cependant, la réponse d'une requête étant un ensemble de motifs, l'utilisateur désire souvent une mise en correspondance de l'intérêt d'un motif extrait par rapport aux autres motifs retournés. Par exemple, un souhait classique est d'éviter la redondance entre motifs. Plus généralement, l'utilisateur désire souvent découvrir des motifs de plus haut niveau reposant sur des caractéristiques qui impliquent plusieurs motifs locaux et donnant au final un sens global à l'ensemble de motifs qui est retourné. Pour cela, l'intérêt d'un motif ne doit pas être considéré de façon indépendante des autres motifs extraits mais il est nécessaire de comparer les motifs entre eux. Remarquons que cette idée est déjà présente dans les représentations condensées de motifs : en effet, les définitions de motifs fermés (cf. définition 1.22) et de motifs libres (cf. définition 1.23) font intervenir deux motifs locaux. Cette voie est aussi une façon de réduire le nombre de motifs extraits et donc de faciliter l'analyse du résultat de la fouille.

Cette section synthétise les deux grandes approches actuelles sur la découverte de motifs ensemblistes dont l'intérêt dépend des autres motifs extraits : d'une part, les *motifs n-aires* [Khiari *et al.*, 2010b] ou *k-pattern sets* [Guns *et al.*, 2013b] et, d'autre part, les *ensembles de motifs* (appelés *pattern sets* dans la littérature anglo-saxonne) [Zimmermann, 2009]. Nous montrons que ces approches se retrouvent dans la notion de *motifs globaux* [Soulet, 2006].

1.4.1 Motifs n-aires

Les *motifs n-aires* [Khiari *et al.*, 2010b, Khiari, 2012] appelés aussi *k-pattern sets* [Guns *et al.*, 2013b] ont été définis dans le but de découvrir des motifs de plus haut niveau. Comme leur nom le suggère, ils tirent leur force de leur capacité à combiner les caractéristiques de plusieurs motifs. Les motifs n-aires sont définis à l'aide de contraintes portant sur n motifs et appelées *contraintes n-aires* ($n \geq 1$, le cas où $n = 1$ correspondant aux motifs locaux). Typiquement, une contrainte n-aire est vérifiée sur l'ensemble du jeu de données.

Définition 1.25. Contrainte n-aire [Khiari *et al.*, 2010b]
Une **contrainte n-aire** est un prédicat $\mathfrak{C} : \mathcal{L}_{\mathcal{I}}^n \rightarrow \{vrai, faux\}$ portant sur n motifs.

Une **requête n-aire** est une conjonction de contraintes contenant au moins une contrainte n-aire.

Définition 1.26. Motif n-aire [Khiari *et al.*, 2010b]
Un **motif n-aire** est un motif défini par une requête n-aire.

Soient un langage de motifs $\mathcal{L}_\mathcal{I}$, un jeu de données $\boldsymbol{r}$ et une contrainte n-aire $\mathfrak{C}$, l'extraction de motifs n-aires consiste à trouver la théorie des motifs n-aires vérifiant :

$$\mathcal{T}h(\mathcal{L}^n_\mathcal{I}, \boldsymbol{r}, \mathfrak{C}) = \{\mathfrak{X} \in \mathcal{L}^n_\mathcal{I} \mid \ \mathfrak{C}(\mathfrak{X}) \ est \ vrai\}$$

Comme $\mathfrak{X} \in \mathcal{L}^n_\mathcal{I}$, on a : $\mathfrak{X} = (\mathtt{x}_1, \ldots, \mathtt{x}_n)$.

Les exemples de motifs n-aires sont très nombreux. Un exemple historique est les règles d'association. Une telle règle implique deux motifs, $\mathtt{x}$ (la prémisse) et $\mathtt{y}$ (la conclusion). Elle est caractérisée par son **support** ($\mathtt{support}(\mathtt{x} \rightarrow \mathtt{y}) = \frac{\mathtt{freq}(\mathtt{x} \cup \mathtt{y})}{|\mathcal{T}|}$) et sa **confiance** ($\mathtt{conf}(\mathtt{x} \rightarrow \mathtt{y}) = \frac{\mathtt{freq}(\mathtt{x} \cup \mathtt{y})}{\mathtt{freq}(\mathtt{x})}$).

Un autre exemple, tirant ses fondements avant même l'apparition de l'expression "fouille de données", est le clustering conceptuel [Fisher, 1987]. Cet exemple fait mieux ressortir la notion de globalité puisque l'un des buts du clustering conceptuel est de couvrir l'ensemble des données. La contrainte n-aire suivante modélise un clustering conceptuel construit à partir de motifs fermés et sans recouvrement :

Exemple 1.7.

$$\{\mathtt{x}_1, \ldots, \mathtt{x}_n \in \mathcal{L}^n_\mathcal{I} \mid \bigwedge_{1 \leq i \leq n} \mathtt{fermé}(\mathtt{x}_i) \ \wedge \bigcup_{1 \leq i \leq n} \mathtt{couverture}(\mathtt{x}_i) = \mathcal{T} \ \wedge \\ \forall 1 \leq i < j \leq n, \mathtt{couverture}(\mathtt{x}_i) \cap \mathtt{couverture}(\mathtt{x}_j) = \emptyset\}$$

D'autres exemples sont les règles d'exception [Suzuki, 2002], les règles inattendues [Padmanabhan et Tuzhilin, 1998], le k-term DNF Learning and Concept Learning [Kearns et Vazirani, 1994], le k-Tiling maximal [Geerts *et al.*, 2004], le redescription Mining [Parida et Ramakrishnan, 2005], les groupes de synexpression [Khiari *et al.*, 2010b], les conflits de classification [Khiari *et al.*, 2010b], etc. Nous détaillons ci-dessous les exemples des règles d'exception et les règles inattendues.

1.4.1-α Règles d'exception [Suzuki, 2002] La notion de *règle d'exception* correspond à une paire de règles composée d'une règle générale et d'une règle déviationnelle par rapport à la règle générale. L'intérêt de la règle déviationnelle repose sur sa mise en correspondance avec la règle générale (contrairement à la notion de règle rare qui est une seule règle peu fréquente et qui est ainsi plus sujette au hasard).

Définition 1.27. Règle d'exception [Suzuki, 2002]
Soient $\mathtt{j}$ un item (par exemple une valeur de classe) et $\mathtt{x}$, $\mathtt{y}$ des motifs locaux, une règle $\mathtt{x} \rightarrow \neg\mathtt{j}$ est une **règle d'exception** si et seulement si il existe $\mathtt{y} \subset \mathtt{x}$ tel que (1) $\mathtt{x} \setminus \mathtt{y} \rightarrow \mathtt{j}$ est une règle fréquente ayant une confiance élevée et (2) $\mathtt{x} \rightarrow \neg\mathtt{j}$ est une règle rare (i.e. peu fréquente) ayant une confiance très élevée.

$\mathtt{x} \setminus \mathtt{y} \rightarrow \mathtt{j}$ est la règle générale et $\mathtt{x} \rightarrow \neg\mathtt{j}$ est la règle d'exception car, généralement, si $\mathtt{x} \setminus \mathtt{y}$ alors $\mathtt{j}$. La règle d'exception isole l'information surprenante. Cette définition suppose que la règle générale a une fréquence élevée et une confiance assez élevée et la règle d'exception a une fréquence faible et une confiance très élevée. Elle illustre aussi la nécessité de combiner plusieurs motifs et donc l'intérêt des motifs n-aires. Précisons que la définition originale [Suzuki, 2002] utilise une troisième règle indiquant

que $\mathtt{y}$ ne détermine pas $\neg\mathtt{j}$ afin de mieux faire ressortir l'apport de l'ajout de $\mathtt{y}$ à $\mathtt{x}$ par rapport à $\mathtt{j}$. Pour simplifier l'exposé, nous omettons cette troisième règle comme cela est fait dans [Khiari, 2012].

Exemple 1.8.
Dans le jeu de données du tableau 1.1 (cf. page 12), la règle de $ACE \rightarrow \neg\mathtt{j}_1$ est une règle d'exception car on a conjointement la règle générale $AC \rightarrow \mathtt{j}_1$ et la règle rare $ACE \rightarrow \neg\mathtt{j}_1$.

Nous illustrons maintenant la modélisation de la notion de règle d'exception avec des contraintes n-aires. Soient $minfr, maxfr, \delta_1, \delta_2 \in \mathbb{N}$ les seuils. La requête n-aire de règle d'exception est formulée comme :

$\mathtt{x} \setminus \mathtt{y} \rightarrow \mathtt{j}$ doit être une règle fréquente ayant une valeur de confiance élevée :	$\rightarrow$	$\begin{array}{c}\mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \geq minfr \\ \wedge \\ \mathtt{freq}(\mathtt{x} \setminus \mathtt{y}) - \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \leq \delta_1\end{array}$
$\mathtt{x} \rightarrow \neg\mathtt{j}$ doit être une règle rare ayant une valeur de confiance très élevée :	$\rightarrow$	$\begin{array}{c}\mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq maxfr \\ \wedge \\ \mathtt{freq}(\mathtt{x}) - \mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq \delta_2\end{array}$

Pour résumer :

$$\mathtt{exception}(\mathtt{x}, \mathtt{y}, \mathtt{j}) \equiv \begin{cases} \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \geq minfr \wedge \\ \mathtt{freq}(\mathtt{x} \setminus \mathtt{y}) - \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \leq \delta_1 \\ \wedge \\ \mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq maxfr \wedge \\ \mathtt{freq}(\mathtt{x}) - \mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq \delta_2 \end{cases}$$

[Suzuki, 2002] propose une méthode fondée sur l'estimation probabiliste pour extraire les règles d'exception, cette méthode est dédiée à ce type de règles.

1.4.1-β Règles inattendues [Padmanabhan et Tuzhilin, 1998]

Dans [Padmanabhan et Tuzhilin, 1998], les auteurs proposent la notion de *règle inattendue*, comme étant une règle qui contredit une croyance préalablement donnée. La méthode extrait l'ensemble des règles $\mathtt{x} \rightarrow \mathtt{y}$ par rapport à une croyance $\mathtt{u} \rightarrow \mathtt{v}$ (où $\mathtt{u}$ et $\mathtt{v}$ sont aussi des motifs). Plus formellement :

Définition 1.28. Règle inattendue [Padmanabhan et Tuzhilin, 1998]

1. $\mathtt{y} \wedge \mathtt{v}$ n'est pas valide,
2. $\mathtt{x} \wedge \mathtt{u}$ est valide ($\mathtt{xu}$ est un motif fréquent),
3. $\mathtt{xu} \rightarrow \mathtt{y}$ est valide ($\mathtt{xu} \rightarrow \mathtt{y}$ est une règle fréquente et de confiance suffisante),
4. $\mathtt{xu} \rightarrow \mathtt{v}$ n'est pas valide (soit $\mathtt{xu} \rightarrow \mathtt{v}$ n'est pas une règle fréquente, soit $\mathtt{xu} \rightarrow \mathtt{v}$ est une règle de faible confiance).

Nous donnons maintenant la modélisation avec des contraintes n-aires de la notion de règle inattendue [Khiari, 2012]. Soient $minfr_1$, $minfr_2$ et $maxfr$ des seuils de fréquence minimale et maximale, $minconf$ et $maxconf$ des seuils de confiance minimale et maximale, une croyance $u \rightarrow v$, on cherche des règles inattendues $\mathtt{x} \rightarrow \mathtt{y}$ telles que :

$$\mathtt{inattendue}(\mathtt{x}, \mathtt{y}) \equiv \begin{cases} \mathtt{freq}(\mathtt{y} \cup \mathtt{v}) = 0 \wedge \\ \mathtt{freq}(\mathtt{x} \cup \mathtt{u}) \geq minfr_1 \wedge \\ \mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{y}) \geq minfr_2 \wedge \\ \mathtt{conf}(\mathtt{x} \cup \mathtt{u} \rightarrow \mathtt{y}) \geq minconf \wedge \\ (\mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{v}) < maxfr \vee \mathtt{conf}(\mathtt{x} \cup \mathtt{u} \rightarrow \mathtt{v}) \leq maxconf) \end{cases}$$

Exemple 1.9.
Dans [Padmanabhan et Tuzhilin, 1998], les auteurs donnent les exemples suivant de règles inattendues :

1. **Croyance :** Les consommateurs avec des enfants ont tendance à acheter des boissons courantes plutôt que des boissons diététiques. **Règle inattendue :** Quand il y a un grand panneau de publicité relié au magasin, les consommateurs ayant des enfants achètent des boissons diététiques.
2. **Croyance :** Les personnes ayant une activité professionnelle ont tendance à acheter plus le week-end qu'en semaine.
   — **Règle inattendue 1 :** En décembre, ces personnes ont tendance à acheter plus en semaine qu'en week-end.
   — **Règle inattendue 2 :** Ces personnes, chargées de familles nombreuses, ont tendance à acheter plus en semaine que le week-end.

La méthode d'extraction des règles inattendues dans [Padmanabhan et Tuzhilin, 1998] est ad hoc à ce type de motifs, tout comme nous avions vu que la méthode des règles d'exception dans [Suzuki, 2002] est spécifique aux règles d'exception. L'extraction de motifs locaux nécessite le parcours d'espaces de recherche qui peuvent être conséquents. Bien évidemment, l'extraction de motifs n-aires requiert des explorations d'espaces encore plus volumineux parce qu'il faut potentiellement comparer les espaces de chaque motif local de la requête (nous revenons sur ce point en section 1.5). Cette grande combinatoire explique certainement l'absence de méthodes génériques jusqu'à très récemment. Cette manque de méthode générique est aussi un frein pour la découverte de motifs de haut niveau car chaque nouveau type de motifs requiert le développement d'une nouvelle méthode. En s'appuyant sur les capacités de modélisation générique et de déclarativité offertes par les méthodes fondées sur la programmation par contraintes, [Khiari *et al.*, 2010b, Guns *et al.*, 2013b] ont proposé des méthodes génériques pour modéliser et extraire les motifs n-aires ou k-pattern sets. Entre autres, les différents exemples que nous avons cités dans cette section sont traités par ces méthodes génériques.

1.4.2 Extraction d'ensembles de motifs fondés sur des contraintes

Afin de capturer l'intérêt de l'ensemble des motifs retournés par une requête, il est naturel de chercher à définir des contraintes portant sur un ensemble de motifs et non plus un seul motif. Dans [Zimmermann, 2009], ces contraintes sont définies comme étant des *contraintes sur des ensembles de motifs.*

Définition 1.29. Contrainte d'ensemble de motifs [Zimmermann, 2009]
Un prédicat $\mathbb{C} : 2^{\mathcal{L}_\mathcal{I}} \rightarrow \{vrai, faux\}$ est une contrainte d'ensemble de motifs.

L'idée générale est analogue à l'extraction de motifs locaux sous contraintes dans le sens où l'information recherchée est toujours caractérisée par des contraintes. Cependant, l'intérêt exprimé par la contrainte porte ici sur un *ensemble* de motifs (par exemple, couverture d'un ensemble de motifs, ensembles de règles, ...). Dans [De Raedt et Zimmermann, 2007], des primitives sont proposées (e.g. support, taille, redondance, représentativité) pour définir les types d'ensembles de motifs recherchés. Ces primitives peuvent être combinées avec des agrégats portant sur un ensemble de motifs (comme `moyenne`, min, max, $\sum$). Un cas particulier de primitives est la comparaison par paire, c'est-à-dire des primitives portant sur des paires de motifs de l'ensemble de motifs (cf. section 1.5 et les motifs optimaux au chapitre 8). La méthode fonctionne en commençant par construire, dans une première phase, le langage correspondant aux ensembles de motifs, puis, dans une deuxième étape, effectue l'extraction à partir de ce nouveau langage. Plus formellement, soient un langage de motifs $\mathcal{L}_\mathcal{I}$, un jeu de données $\boldsymbol{r}$ et une contrainte d'ensemble de motifs $\mathbb{C}$, l'extraction d'ensembles de motifs consiste à trouver la théorie des ensembles de motifs définie par :

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \boldsymbol{r}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \mathbb{C}(\mathbb{X}) \; est \; vrai\}$$

Le résultat est un ensemble d'ensembles de motifs obtenu par formulation de la contrainte $\mathbb{C}$ qui doit être satisfaite pour chaque ensemble de motifs produit (i.e. sous-ensemble de $2^{\mathcal{L}_\mathcal{I}}$). Les auteurs [De Raedt et Zimmermann, 2007] montrent comment les propriétés de monotonie sur l'extraction de motifs locaux peuvent être adaptées à l'extraction d'ensembles de motifs.

Il existe de nombreux exemples de motifs qui peuvent être définis avec des contraintes d'ensembles de motifs. Citons par exemple les Pattern teams [Knobbe et Ho, 2006b], le Tiling minimal [Geerts *et al.*, 2004], the chosen few [Bringmann et Zimmermann, 2007], les top-k motifs [Fu *et al.*, 2000, Han *et al.*, 2002], etc. Nous détaillons ci-dessous trois d'entre eux. D'autre part, le chapitre 8 présente plusieurs autres exemples qui ont la propriété de se modéliser sous la forme de motifs optimaux. Remarquons que les algorithmes d'extraction d'ensembles de motifs sont particuliers à un type de problème. Cependant, la méthode que nous proposons au chapitre 8 peut être vue comme générique dans le sens où elle considère tous les problèmes d'ensembles de motifs qui peuvent être exprimés sous la forme de motifs optimaux.

1.4.2a Pattern teams [Knobbe et Ho, 2006b]

Les **pattern teams** sont un exemple usuel de motifs définis par une contrainte d'ensembles de motifs :

Définition 1.30. Pattern teams [Knobbe et Ho, 2006b]
Soient $\Phi : 2^{\mathcal{L}_\mathcal{I}} \rightarrow \mathcal{R}$ une mesure de qualité et k un entier, la théorie de ce problème est l'ensemble :

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \boldsymbol{r}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \underbrace{\texttt{taille}(\mathbb{X}) = k \wedge (\nexists\, \mathbb{X}' \in 2^{\mathcal{L}_\mathcal{I}} : \texttt{taille}(\mathbb{X}') = k \wedge \Phi(\mathbb{X}') \geq \Phi(\mathbb{X}))}_{\mathbb{C}(\mathbb{X})}\}$$

Les **pattern teams** forment un "petit" sous-ensemble de motifs qui optimisent une mesure de qualité Φ donnée portant sur un ensemble de motifs (cf. la définition 1.11 sur une mesure de qualité). Φ intègre différents aspects comme l'interdépendance entre motifs, le recouvrement entre motifs, la corrélation des motifs envers la classe [Knobbe et Ho, 2006b].

1.4.2b Tiling minimal [Geerts *et al.*, 2004]

Un autre exemple de motifs défini par une contrainte d'ensemble de motifs est le **tiling minimal** [Geerts *et al.*, 2004]. Une tile (i.e. tuile en français) est un motif local auquel on associe la valeur de son aire dans le jeu de données (cf. définition 1.12 page 15). L'aire d'un tiling est le nombre de 1 de toutes les tuiles du tiling, autrement dit la taille de l'union de toutes les tuiles. Le tiling minimal consiste à rechercher un ensemble minimal de tuiles qui couvre tous les 1 du jeu de données.

Définition 1.31. Tiling minimal [Geerts *et al.*, 2004]
Soit $\boldsymbol{r}$ un jeu de données, trouver le tiling de $\boldsymbol{r}$ qui couvre tous les 1 de $\boldsymbol{r}$ et composé du plus petit nombre possible de tuiles.

$$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \boldsymbol{r}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \underbrace{\texttt{couv\_total}(\mathbb{X}) \wedge (\nexists\, \mathbb{X}' \in 2^{\mathcal{L}_\mathcal{I}} : \texttt{couv\_total}(\mathbb{X}') \wedge \texttt{taille}(\mathbb{X}') < \texttt{taille}(\mathbb{X}))}_{\mathbb{C}(\mathbb{X})}\}$$

$$\text{où } \texttt{couv\_total}(\mathbb{X}) \equiv \left(\texttt{aire}(\mathbb{X}) = \sum_{t \in \mathcal{T}, i \in \mathcal{I}} \mathcal{R}(i,t)\right)$$

1.4.2c Les top-k motifs

Les top-k motifs [Fu *et al.*, 2000, Han *et al.*, 2002] sont un problème classique qui consiste à extraire les k meilleurs motifs par rapport à une mesure d'intérêt $\mathtt{m} : \mathcal{L}_{\mathcal{I}} \rightarrow \mathcal{R}$, formellement :

Définition 1.32. TOP-K
Soient $\boldsymbol{r}$ un jeu de données, k un entier et $\mathtt{m} : \mathcal{L}_{\mathcal{I}} \rightarrow \mathcal{R}$ une mesure d'intérêt, trouver les top-k motifs consiste à trouver la théorie :

$$\mathcal{T}h(2^{\mathcal{L}_{\mathcal{I}}}, \boldsymbol{r}, \mathbb{C}) = \{\mathbb{X} \in 2^{\mathcal{L}_{\mathcal{I}}} \mid \underbrace{\forall \mathbb{X}' \in 2^{\mathcal{L}_{\mathcal{I}}} : \mathtt{taille}(\mathbb{X}') \geq k \Rightarrow \forall \mathtt{x} \in \mathbb{X}, \theta(\mathtt{x}) \geq \min_{y \in \mathbb{X}'} \theta(y)}_{\mathbb{C}(\mathbb{X})}\}$$

Exemple 1.10.
Dans le jeu de données du tableau 1.1, les top-2 motifs selon la fréquence sont B ($\mathtt{freq}(B) = 5$) et C ($\mathtt{freq}(C) = 5$). Tous les autres motifs ont une fréquence inférieure à 5.

1.4.3 Contraintes globales

La notion de contraintes globales a été proposée dans [Soulet, 2006, Crémilleux et Soulet, 2008] afin de définir des requêtes portant sur plusieurs motifs locaux et produire des motifs appelés motifs globaux parce qu'ils capturent une propriété portant sur un ensemble de motifs et conférant un caractère de globalité. Plus formellement :

Définition 1.33. CONTRAINTE GLOBALE [CRÉMILLEUX ET SOULET, 2008]
Une contrainte $\mathtt{G}$ est globale si pour vérifier que $\mathtt{G}$ est satisfaite, plusieurs motifs doivent être comparés.

La motivation à l'origine des contraintes globales et des motifs qui en découlent est similaire à celle définissant les motifs n-aires et les ensembles de motifs, à savoir une prise en compte de relations entre motifs et l'expression de propriétés émanant d'un ensemble de motifs. Remarquons que la définition d'une contrainte globale est relativement lâche. Des exemples de motifs issus de contraintes globales sont fournis au tableau 1.4. Comme on le constate, les contraintes globales recouvrent aussi bien les motifs n-aires que les ensembles de motifs. Cette définition des contraintes globales en fait à la fois leur force, parce que l'approche résultante est très générale pour définir des motifs globaux sur les données, avec la possibilité d'intégrer des critères d'optimisation, mais aussi leur faiblesse car la mise en œuvre d'approches génériques pour extraire les motifs ainsi définis est délicate. Ainsi, l'approche générique "approximer/pousser" qui est présentée dans [Soulet, 2006, Crémilleux et Soulet, 2008] nécessite la définition d'une contrainte d'élagage qui réduit de plus en plus l'espace de recherche au cours du processus d'extraction. Des instanciations de cette contrainte ne sont données que pour certains types de motifs, comme par exemple les top-k motifs. On notera que l'idée de contrainte conduisant à successivement réduire l'espace de recherche au cours de l'extraction est similaire à celle des contraintes postées que nous utilisons dans différentes contributions de ce mémoire, marquant ainsi une certaine lignée entre ces travaux.

Il existe d'autres propositions pour la construction de modèles globaux à partir de motifs locaux [Knobbe *et al.*, 2008, Giacometti *et al.*, 2009]. Ces propositions ne définissent pas de méthodes d'extraction de motifs, mais elles permettent de mieux comparer et comprendre celles existantes. Ainsi, [Giacometti *et al.*, 2009] présentent un cadre nommé PBM (Pattern-Based Modeling framework) pour la formalisation et la modélisation de modèles globaux (classifieurs, clustering, résumés) fondés sur des motifs locaux. Dans [Giacometti *et al.*, 2011], les auteurs classifient les contraintes selon leurs liens avec les différents composants de la théorie $\mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \boldsymbol{r}, \mathtt{c})$. Les motifs impliquant plusieurs motifs locaux sont ainsi définis comme étant globaux par rapport à $\mathcal{L}_{\mathcal{I}}$.

Problème		Contrainte globale $\mathtt{G}$
Représentations condensées	Motifs fermés	$\mathtt{fermé(x)} \equiv \begin{cases} vrai & si \quad \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; \mathtt{y} \subset \mathtt{x}, \mathtt{freq(x)} > \mathtt{freq(y)} \\ faux & sinon \end{cases}$
	Motifs libres	$\mathtt{libre(x)} \equiv \begin{cases} vrai & si \quad \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; \mathtt{y} \subset \mathtt{x}, \mathtt{freq(x)} < \mathtt{freq(y)} \\ faux & sinon \end{cases}$
Règles de caractérisation		$\mathtt{caractérisation(x)} \equiv \begin{cases} vrai & si \quad \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; \mathtt{y} \subset \mathtt{x}, \mathtt{x} \rightarrow \mathtt{j} \wedge (\mathtt{y} \rightarrow \neg \mathtt{j}) \\ faux & sinon \end{cases}$
Règles d'exception		$\mathtt{exception(x)} \equiv \begin{cases} vrai & si \quad \exists \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; \mathtt{y} \subset \mathtt{x}, (\mathtt{x} \setminus \mathtt{y} \rightarrow \mathtt{j}) \wedge (\mathtt{x} \rightarrow \neg \mathtt{j}) \\ faux & sinon \end{cases}$
Motifs pic		$\mathtt{pic(x)} \equiv \begin{cases} vrai & si \quad \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; d(\mathtt{x},\mathtt{y}) \leq \delta, \mathtt{m(x)} \geq \varepsilon \times \mathtt{m(y)} \\ faux & sinon \end{cases}$
top-k		$\mathtt{top\text{-}k_m(x)} \equiv \begin{cases} vrai & si \quad \lvert\{\mathtt{y} \in \mathcal{L}_\mathcal{I} \mid \; \mathtt{y} \neq \mathtt{x} \wedge \mathtt{m(y)} > \mathtt{m(x)}\}\rvert < k \\ faux & sinon \end{cases}$

TABLE 1.4 – Contraintes globales

Nous illustrons maintenant la notion de motifs définis à l'aide de contraintes globales avec l'exemple des motifs pic [Crémilleux et Soulet, 2008]. D'autres exemples de motifs définis à l'aide de contraintes globales ont été présentés dans les deux sections précédentes. On notera également que les motifs pics relèvent aussi des ensembles de motifs.

Un motif pic est un motif qui possède une valeur, pour une mesure $\mathtt{m}$ donnée, assez grande par rapport à celles de tous ses voisins :

Définition 1.34. MOTIF PIC
Soient d une distance, $\mathtt{m}$ une mesure, ε un entier et δ un réel, un motif $\mathtt{x}$ est un **motif pic**, si et seulement si pour tous ces voisins $\mathtt{y}$ (i.e. $d(\mathtt{x},\mathtt{y}) \leq \delta$), $\mathtt{m(x)} \geq \varepsilon \times \mathtt{m(y)}$

Les motif pics peuvent être décrits avec la contrainte globale :

$$\mathtt{pic(x)} \equiv \begin{cases} vrai & si \quad \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \; tel \; que \; d(\mathtt{x},\mathtt{y}) \leq \delta, \mathtt{m(x)} \geq \varepsilon \times \mathtt{m(y)} \\ fausse & sinon \end{cases}$$

Exemple 1.11.
Considérons le jeu de données du tableau 1.1 et la mesure de distance $d(\mathtt{x},\mathtt{y}) = |\mathtt{x} \setminus \mathtt{y}| + |\mathtt{y} \setminus \mathtt{x}|$. Le motif BE est un motif pic pour la mesure $\mathtt{aire}$ avec $\delta = 1$ et $\varepsilon = 1,5$. En effet, l'aire de BE (8) surpasse l'aire de tous ces 3 voisins (i.e. $\mathtt{aire}(B) \times \varepsilon = 5 \times 1,5 = 7,5 \leq 8$, $\mathtt{aire}(E) \times \varepsilon = 4 \times 1,5 = 6 \leq 8$ et $\mathtt{aire}(ABE) \times \varepsilon = 3 \times 1,5 = 4,5 \leq 8$).

[Khiari, 2012, Khiari *et al.*, 2012] a certainement défini la première approche pour extraire les motifs pics. Celle-ci est fondée sur les QCSP et la méthode est présentée dans le contexte de l'extraction de motifs n-aires. Nous pensons que ces motifs relèvent plutôt des ensembles de motifs et plus précisément des motifs optimaux (cf. le chapitre 8). La section suivante discute des similarités et différences entre motifs n-aires et ensembles de motifs.

1.5 Discussion et conclusion

Nous dressons dans cette section un bilan des trois approches de motifs fondés sur des motifs locaux présentées à la section précédente. Pour cela, nous discutons de ces approches suivant les dimensions correspondant aux espaces de recherche à parcourir et les méthodes d'extraction mises en œuvre. Nous mettons en relief les caractéristiques propres de chaque approche et nous les comparons les unes par rapport aux autres. Nous situons dans ce paysage notre contribution sur les motifs optimaux présentée au

chapitre 8. Finalement, nous concluons sur l'intérêt d'approches génériques fondées sur la programmation par contraintes.

1.5.1 Espace de recherche

Cette section situe les différentes approches de motifs fondées sur des motifs locaux par rapport à leurs espaces de recherche.

Les ensembles de motifs et les motifs n-aires ne produisent pas le même type de motifs. Comme leur nom l'indique, les ensembles de motifs produisent des *ensembles* (il n'y a pas d'ordre entre les motifs appartenant à un même ensemble de motifs ; un ensemble de motifs ne comporte pas deux (ou plus) occurrences du même motif). En revanche, un motif n-aire correspond à une *séquence de motifs* ou peut aussi être vu comme un n-uplet d'une base de données : les motifs composant un motif n-aire sont ordonnés et un même motif peut être répété. Illustrons ces propos en considérant 3 motifs locaux $\mathtt{x}$, $\mathtt{y}$ et $\mathtt{z}$: $(\mathtt{y},\mathtt{x},\mathtt{z})$ et $(\mathtt{x},\mathtt{y},\mathtt{z})$ forment deux motifs 3-aires distincts ; $(\mathtt{x},\mathtt{x},\mathtt{y})$ est un motif 3-aire, alors qu'il ne peut pas être une solution d'un problème relevant des ensembles de motifs. L'exemple 1.12 - très simple - précise ces différences entre les espaces de recherche.

Exemple 1.12. Ensemble de motifs versus motifs n-aires
Soit $\mathcal{I} = \{A, B\}$. On a : $\mathcal{L}_\mathcal{I} = \{\{A\}, \{B\}, \{AB\}\}$, d'où :
— $2^{\mathcal{L}_\mathcal{I}} = \{\emptyset, \{A\}, \{B\}, \{AB\}, \{A, B\}, \{A, AB\}, \{B, AB\}, \{A, B, AB\}\}$
— $\mathcal{L}^2_\mathcal{I} = \mathcal{L}_\mathcal{I} \times \mathcal{L}_\mathcal{I} = \{(A, A), (A, B), (A, AB), (B, A), (B, B), (B, AB), (AB, A), (AB, B), (AB, AB)\}$

Cette structuration différente des motifs obtenus conduit à des espaces de recherche différents qui sont schématisés à la figure 1.4. $\mathcal{L}_\mathcal{I}$ est un élément de $2^{\mathcal{L}_\mathcal{I}}$ mais pas de $\mathcal{L}^n_\mathcal{I}$ sauf si $n = 1$.

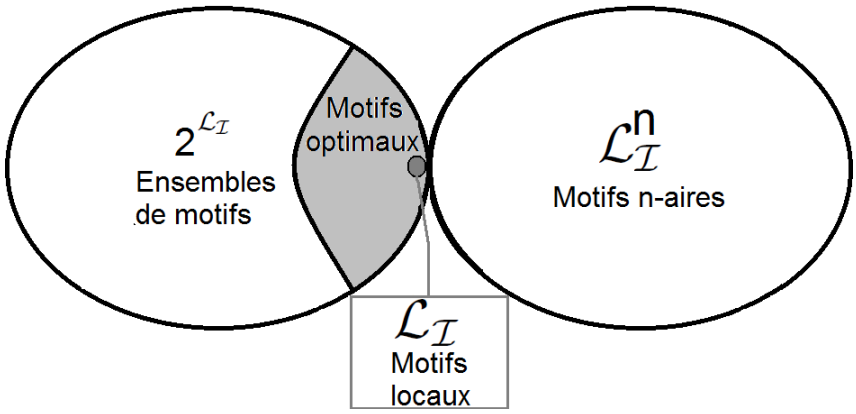


FIGURE 1.4 – Espace de recherche des différents problèmes d'extraction

La table 1.5 donne de façon plus formelle les contraintes et les théories associées à ces espaces de recherche pour un jeu de données $\boldsymbol{r}$.

Contraintes				Théorie		
locale	$\mathtt{c}$:	$\mathcal{L}_\mathcal{I}$	$\rightarrow \{vrai, faux\}$	$\mathcal{T}h(\mathcal{L}_\mathcal{I}, \boldsymbol{r}, \mathtt{c})$	=	$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \mathtt{c}(\mathtt{x})\}$
d'ensembles de motifs	$\mathbb{C}$:	$2^{\mathcal{L}_\mathcal{I}}$	$\rightarrow \{vrai, faux\}$	$\mathcal{T}h(2^{\mathcal{L}_\mathcal{I}}, \boldsymbol{r}, \mathbb{C})$	=	$\{\mathbb{X} \in 2^{\mathcal{L}_\mathcal{I}} \mid \mathbb{C}(\mathbb{X})\}$
n-aire	$\mathfrak{C}$:	$\mathcal{L}^n_\mathcal{I}$	$\rightarrow \{vrai, faux\}$	$\mathcal{T}h(\mathcal{L}^n_\mathcal{I}, \boldsymbol{r}, \mathfrak{C})$	=	$\{\mathfrak{x} \in \mathcal{L}^k_\mathcal{I} \mid \mathfrak{C}(\mathfrak{x})\}$

TABLE 1.5 – Contraintes et théories associées aux différentes approches de motifs

Remarquons que les contraintes globales [Crémilleux et Soulet, 2008] (cf. la définition 1.33) n'ont pas un domaine (i.e. espace de recherche) défini. Ainsi, les contraintes d'ensembles de motifs (cf. la définition 1.29) et les contraintes n-aires (cf. la définition 1.25) sont aussi des contraintes globales (toutes ces contraintes considèrent simultanément plusieurs motifs). Autrement dit, les contraintes globales peuvent être vues comme une subsomption des deux autres types de contraintes.

Comme schématisé à la figure 1.4, les motifs optimaux (cf. le chapitre 8) occupe une place particulière : ils correspondent aux singletons de $2^{\mathcal{L}_\mathcal{I}}$. Cela signifie que ces motifs sont des ensembles de motifs situés à la frontière des motifs locaux. Cette place originale est due à la double nature de ces motifs :

— d'une part, les motifs solutions d'un problème de motifs optimaux forment un *ensemble* de motifs (qui est optimal) par rapport à une préférence (ce point est détaillé au chapitre 8), ces motifs relèvent donc des ensembles de motifs ;

— d'autre part, nous verrons que, étant donnée une préférence, il y a *un unique ensemble de motifs optimaux* qui en forme la solution. Cette unicité de la solution est justement ce qui confère le statut d'optimaux aux motifs de la solution (le problème ayant un unique ensemble comme solution, cet ensemble est vu comme un ensemble optimal de motifs).

La place occupée par les motifs optimaux est remarquable. Comme ils relèvent des ensembles de motifs tout en étant à la frontière des motifs locaux, ces motifs combinent à la fois richesse d'expression et efficacité d'extraction :

— de très nombreux problèmes d'extraction de motifs se modélisent sous la forme de motifs optimaux : citons les motifs libres, fermés, skypatterns, pics, top-k, les tuiles maximales, sous-groupes pertinents, "pattern compression problem", "maximally informative k-itemset", "optimal risk patterns", les "essential jumping emerging pattern", etc. (voir une liste non exhaustive de ces motifs à la section 8.1.2).

— le caractère de globalité des motifs optimaux est assuré par des comparaisons successives entre paires de motifs mais leur extraction est efficace car il est possible d'utiliser des contraintes binaires pour réduire l'espace de recherche ne portant que sur un seul motif inconnu, l'autre motif étant connu. On retrouve alors la situation correspondante aux contraintes locales. Concrètement, la technique va consister à comparer une solution candidate aux solutions déjà obtenues et ajouter de nouvelles contraintes binaires portant sur un seul motif inconnu pour réduire l'espace de recherche.

À l'aide de l'exemple des motifs fermés fréquents, nous illustrons maintenant ces caractéristiques originales des motifs optimaux.

Exemple 1.13. Motifs fermés comme exemple de motifs optimaux
Les motifs fermés fréquents sont un ensemble de motifs optimaux : pour un jeu de données, il n'y a pas deux ensembles de motifs fermés fréquents, autrement dit, l'ensemble de motifs fermés est unique, d'où son optimalité selon la propriété "être fermé". La modélisation des motifs optimaux s'effectue avec un seul motif (comme pour les motifs locaux) mais en y associant une contrainte binaire quantifiée. Ainsi, l'extraction de motifs fermés consiste à extraire $\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \mathtt{fermé(x)}\}$ où la contrainte $\mathtt{fermé(x)}$ n'est pas une contrainte locale mais est définie comme :

$$\mathtt{fermé(x)} =\nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I}, \underbrace{\mathtt{y} \supset \mathtt{x} \wedge \mathtt{freq}(y) = \mathtt{freq(x)}}_{\mathsf{C}(\mathtt{y},\mathtt{x})=\mathtt{y}\rhd\mathtt{x}}$$

ce qui se réécrit en : $\mathtt{fermé(x)} = \forall \mathtt{y} \in \mathcal{L}_\mathcal{I}, \underbrace{\mathtt{y} \supset \mathtt{x} \Rightarrow \mathtt{freq(y)} < \mathtt{freq(x)}}_{\neg\mathsf{C}(\mathtt{y},\mathtt{x})=\mathtt{y}\not\rhd\mathtt{x}}$

Cette contrainte quantifiée se réécrit en :

$$\mathtt{fermé(x)} = \bigwedge_{y \in \mathcal{L}_\mathcal{I}} (y \supset \mathtt{x} \Rightarrow \mathtt{freq}(y) < \mathtt{freq(x)}) = \bigwedge_{y \in \mathcal{L}_\mathcal{I}} \neg\mathsf{C}(y, \mathtt{x})$$

La contrainte $\mathtt{fermé(x)}$ est ainsi réécrite en une conjonction de contraintes binaires.

Les motifs optimaux sont présentés en détail au chapitre 8 et plusieurs exemples seront donnés à la section 8.1.2 (cf. page 136).

Il existe des familles de problèmes pour lesquelles les différents problèmes d'une famille relèvent des différentes approches que nous avons présentées (motifs locaux, ensembles de motifs, motifs n-aires). À titre d'exemple, le tableau 1.6 présente cinq problèmes associés à l'extraction de tuiles [Geerts *et al.*, 2004]. Bien que tous ces problèmes soient liés à l'extraction de tuiles, on constate que, suivant leur définition précise, ils se répartissent dans les différentes approches d'extraction de motifs. Notons que tous ces problèmes sont prouvés NP-difficile [Geerts *et al.*, 2004].

Exemple 1.14. Tiling

N	Problème	Définition	Type
1	**le k-tiling maximal**	Trouver le tiling contenant k tuiles avec une aire maximale.	Motifs n-aires [Guns *et al.*, 2013b]
2	**le tiling minimal**	cf. la définition 1.31.	Ensembles de motifs
3	**l'extraction de grandes tuiles**	Trouver toutes les tuiles avec une aire d'au moins $\psi_{\texttt{aire}}$.	Motifs locaux
4	**les top-k tuiles**	Trouver les k tuiles avec les plus grandes aires.	Motifs optimaux
5	**les tuiles maximales**	Trouver les tuiles avec les plus grandes aires.	Motifs optimaux

TABLE 1.6 – Problèmes associés au tiling

1.5.2 Mise en œuvre des méthodes d'extraction de ces différents types de motifs

L'extraction de motifs locaux est une tâche classique maintenant plutôt bien maîtrisée [Morik *et al.*, 2005]. L'exploitation des contraintes liées à l'(anti-)monotonie et la décomposition de contraintes pour se ramener à des contraintes (anti-)monotones par intervalles ont permis d'importants progrès [Soulet et Crémilleux, 2005, Cerf, 2010].

En ce qui concerne les motifs n-aires, il existe de nombreuses méthodes ad hoc (nous avons présentées celles liées aux règles d'exception [Suzuki, 2002] et aux règles inattendues [Padmanabhan et Tuzhilin, 1998] en section 1.4.1). La programmation par contraintes a permis de développer des méthodes génériques [Khiari *et al.*, 2010b, Guns *et al.*, 2013b] traitant une large gamme de problèmes d'extraction, montrant ainsi la portée de ce type de contraintes et leur application effective.

Si la littérature propose de nombreuses méthodes pour extraire des types particuliers de motifs définis par des contraintes portant sur un ensemble de motifs (cf. section 1.4.2), curieusement, il n'existe pas de méthode générique exceptée [De Raedt et Zimmermann, 2007]. Dans celle-ci, les propriétés connues pour leur intérêt pour l'extraction des motifs locaux (comme l'(anti-)monotonie) sont adaptées à l'extraction d'ensembles de motifs. Mais les expérimentations sont peu nombreuses, cela étant certainement dû à la difficulté intrinsèque d'extraction de ces motifs.

Pour les motifs globaux, [Crémilleux et Soulet, 2008] définissent l'approche *approximer-et-pousser* dont l'idée centrale est de réduire dynamiquement l'espace de recherche au cours du processus d'extraction. Cette approche est appliquée au cas des top-k. D'autres problèmes sont modélisés avec des contraintes globales mais l'instanciation d'*approximer-et-pousser* pour ces problèmes n'est pas indiquée.

1.5.3 Conclusion : vers des approches génériques fondées sur la programmation par contraintes

Dans ce chapitre, nous avons présenté la problématique de l'extraction de motifs et proposé une synthèse des approches d'extraction d'ensembles de motifs fondés sur des motifs locaux. Depuis l'origine de la formalisation des problèmes d'extraction de motifs sous contraintes [Mannila et Toivonen, 1997], la recherche de méthodes de modélisation et de résolution génériques est un des objectifs de la communauté. Cependant, dans la pratique, et encore plus particulièrement dans le domaine des ensembles de motifs, beaucoup de méthodes restent dédiées à des types particuliers de motifs.

Un virage s'est amorcé il y a quelques années avec les premiers travaux conjoints entre fouille de données et programmation par contraintes (PPC). Le but principal est d'offrir à l'utilisateur des méthodes déclaratives, génériques et plus expressives pour modéliser les problèmes qu'il souhaite résoudre. L'utilisateur n'a alors plus à se préoccuper de l'étape de résolution, celle-ci étant prise en charge par la "machinerie" de la PPC. Notre travail s'inscrit dans ce courant récent, marqué par exemple par les contributions [Khiari, 2012, Guns, 2012, De Raedt *et al.*, 2008, Khiari *et al.*, 2010b] pour introduire la généricité dans les méthodes d'extraction de motifs en utilisant la PPC. Parmi les contributions présentées dans ce mémoire, nous pensons que celle sur les motifs optimaux est la plus empreinte du caractère de généricité. L'utilisation de la PPC est au cœur de nos travaux et le chapitre suivant en synthétise les principales notions.

"Des contraintes naît la beauté."

Léonard de Vinci

Chapitre 2

Problèmes de Satisfaction de Contraintes

Sommaire

Ce chapitre introductif regroupe les notions de PPC dont nous aurons besoin tout au long de ce mémoire. Ce chapitre présente, tout d'abord, les notions de base relatives aux CSPs : cohérence, filtrage et méthodes de recherche. Puis, nous nous intéressons successivement :

- aux contraintes réifiées qui seront utilisées pour modéliser les problèmes d'extraction de motifs (cf. le chapitre 3 à la page 45),

- à la relaxation de contraintes en PPC sur laquelle reposeront les contraintes souples de seuil (cf. le chapitre 5 à la page 59),

- aux CSP dynamiques qui permettront l'extraction des skypatterns (cf. le chapitre 6 à la page 85), favoriseront la construction des cubes de skypatterns (cf. le chapitre 7 à la page 111), ainsi que l'extraction des motifs optimaux (cf. le chapitre 8 à la page 135).

2.1 Définitions

Définition 2.1. CSP
Un CSP est un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ tel que : - $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ est l'ensemble des **variables**,

- $\mathcal{D} = \{D_{x_1}, D_{x_2}, \ldots, D_{x_n}\}$ est l'ensemble des **domaines des variables**. Chaque domaine D_{x_i} est un ensemble fini contenant les valeurs possibles pour la variable x_i.

- $\mathcal{C} = \{c_1, c_2, \ldots, c_e\}$ est l'ensemble des **contraintes**. Chaque contrainte c_i porte sur un sous-ensemble de variables de $\mathcal{X}$ appelé $var(c_i)$. L'arité (nombre de variables) de c_i est définie par $|var(c_i)|$.

Définition 2.2. Instanciation
Une **instanciation élémentaire** I est un couple variable-valeur (x_i, v) tel que $v \in D_{x_i}$. Une **instanciation** est un ensemble d'instanciations élémentaires. Si une instanciation porte sur toutes les variables alors on parle d'**instanciation complète**, autrement on parle d'**instanciation partielle**.

Définition 2.3. Solution
Une **solution** d'un CSP est une instanciation complète S telle que toutes les contraintes de P sont satisfaites par S.

Exemple 2.1.
On considère le CSP suivant :
- $\mathcal{X} = \{x_1, x_2, x_3\}$,
- $\mathcal{D} = \{D_{x_1}, D_{x_2}, D_{x_3}\}$ avec $D_{x_1} = D_{x_2} = D_{x_3} = \{1, 2, 3\}$,
- $\mathcal{C} = \{c_1, c_2\}$ avec $c_1 = [x_1 < x_2]$ et $c_2 = [x_2 = x_3]$.

Alors :

$var(c_1) = \{x_1, x_2\}$

$I_1 = \{(x_1, 1), (x_2, 2)\}$, $I_2 = \{(x_1, 1), (x_2, 3)\}$ et $I_3 = \{(x_1, 2), (x_2, 3)\}$ sont trois instanciations partielles satisfaisant c_1.

$var(c_2) = \{x_2, x_3\}$

$I'_1 = \{(x_2, 1), (x_3, 1)\}$, $I'_2 = \{(x_2, 2), (x_3, 2)\}$ et $I'_3 = \{(x_2, 3), (x_3, 3)\}$ sont trois instanciations partielles satisfaisant c_2.

Il existe trois solutions (instanciations complètes satisfaisant c_1 et c_2) :
- $S_1 = \{(x_1, 1), (x_2, 2), (x_3, 2)\}$,
- $S_2 = \{(x_1, 1), (x_2, 3), (x_3, 3)\}$,
- $S_3 = \{(x_1, 2), (x_2, 3), (x_3, 3)\}$

2.2 Filtrage par Arc-Cohérence

Les CSP sont le plus souvent résolus par des méthodes de recherche arborescente où des sous-arbres sont élagués par filtrage. Le filtrage consiste à identifier certaines valeurs ne pouvant apparaître dans une solution et à les retirer des domaines correspondants. Les sous-arbres associés ne seront alors pas développés par l'algorithme de recherche. Ainsi, le filtrage permet d'obtenir un nouveau CSP équivalent au CSP de départ (ils possèdent exactement les mêmes solutions), mais dont la taille de l'arbre de recherche sera plus petite.

Enfin, les algorithmes de filtrage doivent être de complexité polynomiale. En effet, le temps gagné (en évitant de parcourir certains sous-arbres inutiles) doit être supérieur au temps nécessaire pour réaliser les filtrages successifs. Le filtrage le plus communément utilisé est le filtrage par Arc-Cohérence (AC). Sa

complexité temporelle, dans le pire cas, est en $\mathcal{O}(e \times d^2)$, avec e nombre de contraintes et d taille du plus grand domaine.

Définition 2.4. Support d'une valeur pour une contrainte
Étant donné une contrainte binaire c telle que : $var(c) = \{x_1, x_2\}$. La valeur (x_j, v_j) est un **support** de la valeur (x_i, v_i) pour la contrainte c si et seulement si (v_i, v_j) satisfait c.

Définition 2.5. Viabilité d'une valeur
Une valeur est **viable** si et seulement si elle possède au moins un support pour chacune des contraintes portant sur elle.

Définition 2.6. Arc-Cohérence d'un CSP
Un CSP est arc-cohérent si et seulement si aucun domaine n'est vide et si toutes les valeurs des domaines sont viables.

Exemple 2.2.
Le filtrage par arc-cohérence du CSP de l'exemple 2.1 conduit aux domaines suivants :
- $D_{x_1} = \{1, 2\}$,
- $D_{x_2} = \{2, 3\}$,
- $D_{x_3} = \{2, 3\}$.

Algorithme 1 : Revise (cas des contraintes binaires)

Données : x_i : variable, x_j : variable, D_{x_i} : domaine de x_i
Résultat : éliminé :booléen

```
1 éliminé ← Faux;
2 pour tous les v ∈ D_{x_i} faire
3     si ∄ v_j ∈ D_{x_j} t.q. (v, v_j) satisfait c_i alors
4         D_{x_i} ← D_{x_i}\{v};
5         éliminé ← Vrai;
6 retourner éliminé ;
```

De nombreux algorithmes de filtrage ont été proposés pour rendre un CSP arc-cohérent. AC-3 fut l'une des premières méthodes proposées [Mackworth, 1977]. L'algorithme Revise (x_i, x_j, D_{x_i}) (cf. l'algorithme 1) retire du domaine D_{x_i} de la variable x_i les valeurs sans support pour la contrainte $c_{i,j}$. La complexité temporelle de Revise est en $\mathcal{O}(d^2)$ dans le pire cas.

AC-3 (cf. l'algorithme 2) gère un ensemble Q de contraintes à traiter. Initialement, Q contient tous les couples $(c_{i,j}, x_i)$. Tant que Q n'est pas vide, un même traitement est effectué : le premier couple $(c_{i,j}, x_i)$ de Q est sélectionné ; puis, on tente de réduire le domaine D_{x_i} de la variable x_i à l'aide de l'algorithme Revise. Si D_{x_i} a été modifié par Revise, alors tous les couples $(c_{i,k})$, avec $k \neq j$, non présents dans Q, sont ajoutés. Si le domaine d'une variable devient vide, alors AC-3 retourne `Echec` signalant l'absence de solution. Lorsque Q devient vide, l'algorithme s'arrête et le CSP est arc-cohérent. AC-3 maintient la cohérence d'arc en $\mathcal{O}(e \times d^3)$.

Plusieurs améliorations de AC-3 ont été proposées :
- AC-4 [Mohr et Henderson, 1986] a été la première méthode permettant d'établir l'AC en temps optimal dans le pire cas, i.e. en $\mathcal{O}(e \times d^2)$. AC-4 calcule, tout d'abord, tous les supports de toutes les valeurs, puis effectue le filtrage proprement dit en $\mathcal{O}(e \times d^2)$.
- AC-6 [Bessière et Cordier, 1993] a permis de diminuer la complexité en moyenne en gérant de manière paresseuse les supports. Au lieu de calculer l'ensemble de tous les supports pour toutes les valeurs, AC-6 s'assure seulement qu'il existe au moins un support pour chaque valeur. Lorsqu'un

Algorithme 2 : AC-3 (cas des contraintes binaires)

Données : $\mathcal{X}$: ensemble de variables, $\mathcal{D}$: ensemble de domaines, $\mathcal{C}$: ensemble de contraintes
Résultat : $\{\texttt{Echec}, \texttt{Succes}\}$

```
1 Soit Q = {(c_{i,j}, x_i) t.q. c_{i,j} ∈ C, x_i ∈ X}
2 tant que Q ≠ ∅ faire
3     Extraire (c_{i,j}, x_i) de Q;
4     si REVISE (x_i, x_j, D_{x_i}) alors
5         si D_{x_i} = ∅ alors
6             retourner Echec;
7         Q ← Q ∪ {(c_{k,i}, x_k) t.q. k ≠ j};
8 retourner Succes;
```

tel support est retiré AC-6 ne recherche plus un nouveau support à partir du début du domaine, mais à partir de celui venant d'être retiré.

— AC-2001 [Bessière et Régin, 2001] a encore amélioré la complexité en moyenne en stockant, dans une structure de données additionnelle, le dernier support obtenu et en testant, toujours en premier, la présence de celui-ci.

2.3 Résolution

Les méthodes de recherche arborescente reposent généralement sur le principe du *backtrack* [Golomb et Baumert, 1965] associé à des méthodes de filtrage permettant d'élaguer l'arbre de recherche. De telles méthodes de recherche arborescente pour les CSP sont sûres (on n'obtient que des solutions) et complètes (on obtient toutes les solutions). Dans cette section, nous rappelons le parcours fondé sur le retour arrière chronologique ainsi que le parcours avec Maintien de l'Arc-Cohérence (MAC [Sabin et Freuder, 1994, Bessière et Régin, 1996]) qui réalise un filtrage par AC à chaque nœud de l'arbre.

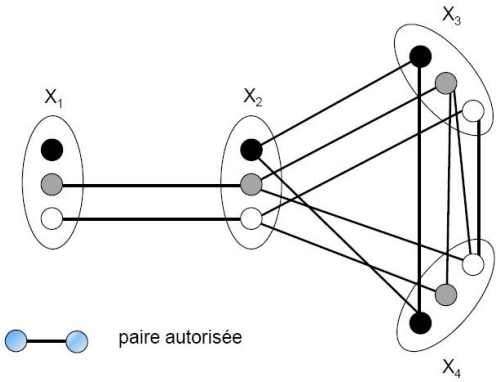


FIGURE 2.1 – Graphe de cohérence

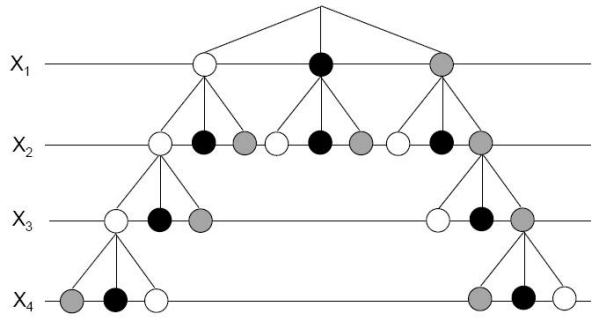


FIGURE 2.2 – Arbre de recherche parcouru par backtrack

2.3.1 Backtrack chronologique

Les variables du problème sont instanciées avec les valeurs de leur domaine, dans un ordre prédéfini, jusqu'à ce que l'un de ces choix viole une contrainte. La dernière instanciation effectuée est alors remise en cause. Une nouvelle valeur est essayée pour la dernière variable instanciée (variable courante). Si toutes les valeurs du domaine de cette variable ont été visitées, on doit procéder à un retour en arrière. Pour cela, une nouvelle valeur est choisi pour la variable précédant immédiatement la variable courante. Ce processus est répété jusqu'à obtenir une solution, i.e. une instanciation de toutes les variables ne violant aucune contrainte. Si on a parcouru tout l'arbre de recherche sans en trouver, alors le problème ne possède aucune solution.

Exemple 2.3.
Soit le CSP décrit par son graphe de cohérence [11] à la figure 2.1 [12]. L'arbre de recherche par Backtrack chronologique est décrit par la figure 2.2.

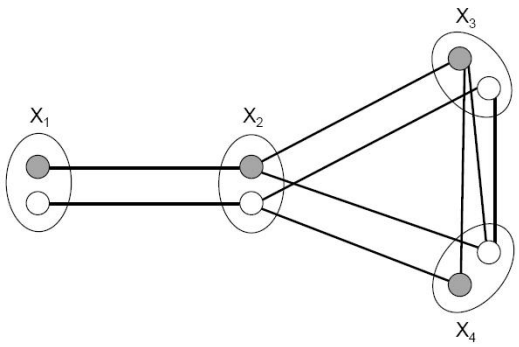


FIGURE 2.3 – Graphe de cohérence après filtrage par AC

2.3.2 Maintien d'arc-cohérence

Supposons qu'en cherchant une solution, Chronological Backtracking donne à une variable x_i une valeur v qui exclut toutes les valeurs possibles pour une variable x_j. L'algorithme Chronological Backtra-

11. le graphe de cohérence est un graphe dont les sommets sont les valeurs des domaines, et dont les arcs relient les sommets représentant des valeurs compatibles.
12. Ce CSP n'ayant qu'une unique solution (gris, gris, gris, blanc), on a choisi un ordre de parcours des valeurs de manière à explorer l'intégralité de l'arbre de recherche pour trouver cette solution.

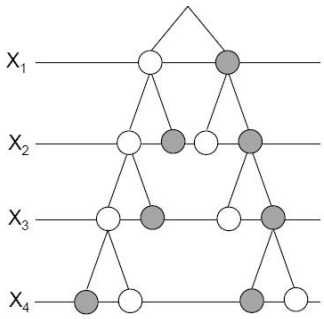


FIGURE 2.4 – Arbre de recherche parcouru par MAC

cking ne s'en rendra compte que lorsque x_j sera considérée. De plus, avec un retour arrière chronologique, avant que x_i ne soit reconsidérée, il se peut qu'une grande partie de l'espace de recherche soit explorée en vain. Ceci pourrait être évité en prenant en compte le fait que (x_i, v) ne pourra jamais faire partie d'une solution, puisqu'il n'y a aucune valeur de x_j qui soit compatible avec (x_i, v). Utiliser le filtrage permet d'éviter la situation décrite ci-dessus. Une valeur v n'est acceptée qu'après avoir vérifié les domaines des variables futures. S'il existe un domaine qui ne contient que des valeurs incompatibles avec v, alors v est éliminée. Par ailleurs, le filtrage peut être utilisé après chaque instanciation (x_i, v). En effet, les valeurs des variables futures qui ne sont pas compatibles avec (x_i, v) peuvent être éliminées sans perte de solutions.

Les algorithmes les plus connus sont : Forward Checking [Haralick et Elliott, 1980] et Maintaining Arc Concistency (MAC) [Sabin et Freuder, 1994, Bessière et Régin, 1996]. L'algorithme de maintien d'arc cohérence MAC procède au au filtrage par AC après chaque instanciation de variable.

Algorithme 3 : *Depth-First*-1(D)

```
1 D ← Filter(D, C);
2 si il existe x_i ∈ 𝒳 tel que D_{x_i} est vide alors
3 |  retourner Échec;
4 si il existe x_i ∈ 𝒳 tel que |D_{x_i}| > 1 alors
5 |  Sélectionner x_i ∈ 𝒳 tel que |D_{x_i}| > 1;
6 |  pour tous les v ∈ D_{x_i} faire
7 |  |  Depth-First-1(D ∪ {x_i → {v}});
8 sinon
9 |  Output(D, C);
```

L'algorithme 3 montre comment un CSP peut être résolu en utilisant une recherche en profondeur d'abord. D et C désignent respectivement l'ensemble des domaines courants et l'ensemble courant de contraintes. $Filter(D, C)$ effectue le filtrage des domaines de D selon les contraintes de C (ligne 1). A chaque nœud de l'arbre de recherche, l'algorithme fait le branching en instanciant une variable par une valeur de son domaine courant (lignes 6 et 7). Il revient en arrière en cas de violation d'une des contraintes, i.e. lorsqu'au moins un domaine est vide (ligne 2). La recherche peut être améliorée en choisissant avec soin [13] la variable qui sera affectée (ligne 5). On obtient une solution (ligne 9) lorsque chaque domaine D_{x_i} est réduit à un singleton et toutes les contraintes sont satisfaites.

13. par exemple, en utilisant l'heuristique *dom/deg* qui sélectionne la variable de plus petit ratio entre la taille de son domaine courant et le nombre de contraintes où elle figure.

2.4 Contraintes réifiées

Une contrainte réifiée associe une variable x de domaine $\{0,1\}$ à une contrainte c de manière à refléter la satisfaction de la contrainte (valeur 1), ou sa non-satisfaction (valeur 0).

$$(x = 1) \iff c \quad avec \quad D_x = \{0,1\} \quad et \quad x \notin var(c)$$

De telles contraintes sont très utiles pour exprimer des formules booléennes sur des contraintes et pour exprimer qu'un certain nombre de contraintes doivent être satisfaites.

Le filtrage des contraintes réifiées s'effectue comme suit :

1. Pour l'implication :
   — si la contrainte $x = 1$ est satisfaite alors la contrainte c doit être satisfaite,
   — si la contrainte $x = 0$ est satisfaite alors la contrainte $\neg c$ doit être satisfaite,
2. Pour la deuxième implication :
   — si la contrainte c est satisfaite alors la contrainte $x = 1$ doit être satisfaite,
   — si la contrainte c n'est pas satisfaite alors la contrainte $x = 0$ doit être satisfaite.

Les contraintes réifiées seront utilisées au chapitre 3 (cf. page 45) pour modéliser les problèmes d'extraction de motifs et notamment pour réaliser l'encodage booléen de ces motifs (cf. la section 3.2 à la page 47).

2.5 Relaxation de contraintes

De nombreux problèmes réels sont par nature sur-contraints (ils ne possèdent aucune solution). Cela est souvent dû à un ensemble d'exigences trop fortes des utilisateurs de l'application développée. Ainsi, dans les problèmes d'emplois du temps, les souhaits (préférences) exprimés par les élèves, les enseignants et l'administration sont souvent incompatibles. Différents modèles ou cadres pour la relaxation des CSPs ont été proposés : les CSPs hiérarchiques [Borning *et al.*, 1992], les Partial CSPs [Freuder et Wallace, 1992], les CSPs valués [Schiex *et al.*, 1995], les Semi-ring CSPs [Bistarelli *et al.*, 1999], la relaxation disjonctive [Petit, 2002], ...

Le cadre que nous avons retenu, pour modéliser l'extraction de motifs sous contraintes souples de seuil (cf. le chapitre 5 à la page 59), est le modèle disjonctif proposé par Thierry Petit [Petit, 2002]. Dans cette section, nous présentons successivement la problématique, le modèle disjonctif et les raisons qui ont motivé notre choix.

2.5.1 Problématique

Relaxer une contrainte (quelconque) c'est l'autoriser à ne pas être nécessairement satisfaite contre un coût. Lorsque l'on cherche à modéliser un problème sur-contraint, on distingue deux catégories de contraintes :

1. Les *contraintes d'intégrité* (ou contraintes dures) qui doivent être impérativement satisfaites. Les contraintes d'intégrité reflètent généralement des obligations d'ordre physique.
   Exemples : un enseignant ne peut faire deux cours en même temps, une salle ne peut accueillir deux cours en même temps, ...

2. Les *contraintes de préférence* (ou contraintes souples) qui expriment des souhaits formulés sous forme de propriétés que l'on aimerait voir vérifiées par une solution. Les contraintes de préférence permettent de spécifier les propriétés que devrait satisfaire une solution jugée de bonne qualité. Exemples : certains enseignants souhaitent ne pas faire plus de six heures de cours par jour, les étudiants souhaitent que les cours commencent après 9 heures, ...

Le but de la relaxation n'est plus de satisfaire toutes les contraintes (cela n'est pas toujours possible), mais de les satisfaire *au mieux*, i.e. satisfaire toutes les contraintes dures et minimiser une agrégation des coûts des contraintes souples insatisfaites. Ainsi, la relaxation se modélise sous forme d'un COP (Problème d'Optimisation sous Contraintes).

2.5.2 Relaxation disjonctive d'un CSP

Thierry Petit et al. proposent dans [Petit, 2002, Petit *et al.*, 2000] de transformer un COP en un problème de satisfaction afin de bénéficier du savoir faire important déjà disponible pour résoudre les CSPs. Pour cela, les coûts sont intégrés directement au problème via l'ajout d'un ensemble de variables. dites variables de coût.

Définition 2.7. Variable de coût associée à une contrainte [Petit, 2002]
Étant donnée une contrainte c_i, la variable de coût z_i (associée à c_i) est une variable à valeurs numériques positives ou nulle telle que :
— si c_i est satisfaite alors $z_i = 0$,
— si c_i est insatisfaite alors $z_i > 0$.
La valeur de z_i quantifie la violation selon la sémantique de violation retenue pour la contrainte c_i.

Définition 2.8. Sémantique de violation
μ est une sémantique de violation pour la contrainte $c(x_1, \ldots, x_n)$ si et seulement si μ est une fonction de $D_{x_1} \times \ldots \times D_{x_n}$ vers $\mathbb{R}^+$ telle que : $\forall \mathcal{A} \in D_{x_1} \times \ldots \times D_{x_n}, \mu(\mathcal{A}) = 0$ si et seulement si $c(x_1, \ldots, x_n)$ est satisfaite.

Exemple 2.4.
Soit la contrainte binaire c_i définie par $x_1 = x_2$, avec pour domaines $D_{x_1} = D_{x_2} = \{1, 2, 3\}$. Si l'on choisit comme sémantique de violation la distance entre les deux variables, alors $z_i = |x_1 - x_2|$. Les coûts de violation de différentes instanciations sont présentés dans les tableaux suivants :

X_1	X_2	z_i
1	1	0
1	2	1
1	3	2

X_1	X_2	z_i
2	1	1
2	2	0
2	3	1

X_1	X_2	z_i
3	1	2
3	2	1
3	3	0

Figure 2.5 – Coûts de violation pour l'exemple 2.4

La version relaxée de chaque contrainte est formulée sous forme d'une disjonction : soit la contrainte est vérifiée et le coût est nul, soit la contrainte est insatisfaite et le coût est précisé.

Définition 2.9. Relaxation disjonctive d'une contrainte [Petit, 2002]
Étant donné un CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, une contrainte $c_i \in \mathcal{C}$, sa négation $\bar{c_i}$ et sa variable de coût z_i. La relaxation disjonctive de c_i est la contrainte c_{disj_i} définie par :

$$c_{disj_i} \equiv [c_i \wedge z_i = 0] \vee [\bar{c_i} \wedge z_i > 0]$$

Exemple 2.5.
La relaxation disjonctive de la contrainte c_i de l'exemple 2.4 est la suivante :

$$c_{disj_i} \equiv [x_1 = x_2 \wedge z_i = 0] \vee [x_1 \neq x_2 \wedge z_i = |x_1 - x_2|]$$

Soit C_s l'ensemble des contraintes souples et C_h l'ensemble des contraintes dures. A chaque contrainte $c_i \in C_s$, on associe z_i qui est sa variable de coût. Soit z la variable représentant la violation totale (cumul des violations), alors $z = \sum_{c_i \in C_s} z_i$. Soit λ la quantité maximale de violation que l'on s'autorise ; alors, on doit avoir : $z \leq \lambda$, i.e. $\sum_{c_i \in C_s} z_i \leq \lambda$. Le domaine de z est l'intervalle $[0..\lambda]$.

Définition 2.10. Relaxation disjonctive d'un CSP [Petit, 2002]
Étant donné un CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, la relaxation disjonctive de P est CSP $P' = (\mathcal{X}', \mathcal{D}', \mathcal{C}')$ dérivé de P tel que :
- $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}_\mathcal{Z} \cup \{z\}$
- $\mathcal{D}' = \mathcal{D} \cup D_\mathcal{Z} \cup \{[0..\lambda]\}$
- $\mathcal{C}' = \mathcal{C}_h \cup \mathcal{C}_{disj} \cup \{z = \bigoplus_{i=1}^{|C_{disj}|} z_i\}$

avec
- $\mathcal{X}_\mathcal{Z} = \{z_1, \ldots, z_{|C_s|}\} \cup \{z\}$, l'ensemble des variables de coût,
- $D_\mathcal{Z} = \{D_{z_1}, \ldots, D_{z_{|C_s|}}\} \cup \{[0..\lambda]\}$, l'ensemble des domaines des variables de coût,
- C_{disj} l'ensemble des contraintes souples exprimées sous forme disjonctive,
- C_h l'ensemble des contraintes dures.

Exemple 2.6.
Soit le CSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ tel que :
- $\mathcal{X} = \{x_1, x_2, x_3\}$,
- $\mathcal{D} = \{D_{x_1} = \{2, 3\}, D_{x_2} = \{1, 2\}, D_{x_3} = \{1, 2, 3\}\}$,
- $\mathcal{C} = \{c_1 = [x_1 > x_2], c_2 = [x_1 = x_3], c_3 = [x_3 > x_2]\}$

On suppose que :
- c_1 est une contrainte dure.
- c_2 et c_3 sont des contraintes souples.
- La sémantique de violation de c_2 est la distance entre les valeurs de ses deux variables.
- La sémantique de violation de c_3 est le carré de l'écart entre les valeurs de ses deux variables.

La relaxation disjonctive de P est le CSP $P' = (\mathcal{X}', \mathcal{D}', \mathcal{C}')$ tel que :

- $\mathcal{X}' = \mathcal{X} \cup \{z_1, z_2, z\}$
- $\mathcal{D}' = \mathcal{D} \cup \{D_{z_1}, D_{z_2}, D_z\}$
- $\mathcal{C}' = \{x_1 > x_2,$
  $[x_1 = x_3 \wedge z_1 = 0] \vee [x_1 \neq x_3 \wedge z_1 = |x_1 - x_3|],$
  $[x_3 > x_2 \wedge z_2 = 0] \vee [x_3 \leq x_2 \wedge z_2 = (x_3 - x_2)^2]$
  $z = z_1 + z_2\}$

Remarque : Cette forme générale est difficilement exploitable en raison des disjonctions apparues. Dans certains cas, on peut simplifier cette écriture de manière drastique.

2.5.3 Choix du modèle disjonctif pour les contraintes souples de seuil

Le modèle disjonctif de Thierry Petit est le cadre que nous avons retenu pour modéliser l'extraction de motifs sous contraintes souples de seuil (cf. le chapitre 5 à la page 59).

Tout d'abord, les variables de coût permettent de modéliser la violation de manière simple : équilibrage de la violation, limitation de la violation d'une contrainte, ou encore ajout de contraintes sur les variables de coût. De plus, l'ajout de contraintes portant sur les variables initiales et les variables de coût (méta-contraintes) permet de contrôler de manière fine la violation. Enfin, grâce au filtrage, les méta-contraintes permettent de filtrer les domaines des variables de coût et ainsi provoquer le filtrage des domaines des variables initiales du problème.

Le formalisme des Weighted CSP (WCSP) aurait constitué une solution alternative. Mais, dans cette phase exploratoire, nous ne les avons pas retenus car les solveurs de WCSPs ne savent prendre en compte que les contraintes binaires ou ternaires. Pour une comparaison plus détaillée entre les WCSPs et le modèle disjonctif, nous conseillons au lecteur de se reporter à la thèse de Jean-Philippe Métivier [Métivier, 2010].

Pour conclure, le modèle disjonctif possède à la fois la possibilité d'exprimer des mesures de violation fines et d'exprimer facilement des règles de contrôle de la violation (notamment grâce aux variables de coût). De plus ce modèle ne possède pas de limitation sur les contraintes utilisables (ni sur le nombre de variables, ni sur la nature de la contrainte). Les bonnes propriétés du modèle disjonctif nous ont conduit à le retenir pour modéliser l'extraction de motifs sous contraintes souples de seuil (cf. le chapitre 5 à la page 59).

2.6 CSP dynamiques

Un CSP dynamique (DCSP) [Verfaillie et Jussien, 2005] est une séquence $P_1, P_2, \ldots, P_n$ de CSP, où chaque CSP P_i résulte de changements apportés au précédent P_{i-1}. Ces changements peuvent affecter les variables, les domaines et les contraintes.

Dans notre mémoire, les seuls changements effectués sont l'ajout de nouvelles contraintes. Résoudre un CSP dynamique consiste alors à résoudre un seul CSP avec des contraintes supplémentaires postées pendant la recherche. Chaque fois qu'une nouvelle solution est trouvée, de nouvelles contraintes sont ajoutées. Ces contraintes vont survivre au retour en arrière et imposent que les prochaines solutions doivent vérifier à la fois l'ensemble courant de contraintes ainsi que les contrainte postées.

L'algorithme 4 ci-dessous décrit la résolution d'un CSP dynamique avec ajout de contraintes $\phi(\mathcal{X})$ portant sur l'ensemble $\mathcal{X}$ des variables du CSP. Cet algorithme est identique à l'algorithme 3, si ce n'est qu'une fois la première solution obtenue, le nouvel ensemble de contraintes à satisfaire est l'ancien auquel on a ajouté $\phi(\mathcal{X})$ (ligne 10). Enfin, C est une variable qui est globale par rapport à l'algorithme 4 *Depth-First*(D) 2.

Algorithme 4 : *Depth-First*-2(D)

```
1  D ← Filter(D, C);
2  si il existe x_i ∈ X tel que D_{x_i} est vide alors
3  |  retourner Échec;
4  si il existe x_i ∈ X tel que |D_{x_i}| > 1 alors
5  |  Sélectionner x_i ∈ X tel que |D_{x_i}| > 1;
6  |  pour tous les v ∈ D_{x_i} faire
7  |  |  Depth-First-2(D ∪ {x_i → {v}});
8  sinon
9  |  Output(D, C);
10 |  C ← C ∪ {φ(X)};
```

Remarque : Si, lorsque l'on ajoute dynamiquement une contrainte, il suffit de relancer le filtrage, il n'en serait pas de même s'il s'agissait du retrait dynamique d'une contrainte. En effet, dans ce cas, il faut pouvoir restaurer, dans les domaines de toutes les variables, toutes les valeurs dont la contrainte retirée a provoqué le retrait, que ce soit directement (par elle-même), ou indirectement (en conjonction avec d'autres contraintes). Différents algorithmes ont été proposés pour pouvoir réaliser efficacement les retraits dynamiques de contraintes. Pour plus de détails, nous conseillons au lecteur de se reporter à : [Ginsberg, 1993], [Debruyne, 1996], [Jussien *et al.*, 2000].

2.7 Conclusion

Dans ce chapitre, nous avons regroupé les principales notions de PPC dont nous aurons besoin tout au long de ce mémoire. Après avoir brièvement présenté les CSPs, nous nous sommes successivement intéressés :
- aux contraintes réifiées utilisées pour modéliser les problèmes d'extraction de motifs (cf. le chapitre 3 à la page 45),
- à la relaxation de contraintes en PPC sur laquelle reposeront les contraintes souples de seuil (cf. le chapitre 5 à la page 59),
- aux CSP dynamiques qui permettront l'extraction des skypatterns (cf. le chapitre 6 à la page 85), favoriseront la construction des cubes de skypatterns (cf. le chapitre 7 à la page 111), ainsi que l'extraction des motifs optimaux (cf. le chapitre 8 à la page 135).

"Il y a, frères, énormément à faire !"

César Vallejo, Les neuf monstres

Chapitre 3

Extraction de motifs modélisée sous forme d'un CSP

Sommaire

Ce chapitre s'intéresse à la modélisation, sous forme de CSP, des problèmes d'extraction de motifs sous contraintes. Tout d'abord, nous décrivons le cadre général et nous l'illustrons en prenant l'exemple des règles d'exception. Puis, nous nous intéressons de manière détaillée à l'encodage booléen du ou des motif(s) recherché(s). La modélisation d'une requête sous forme d'un CSP sera plus particulièrement utilisée au chapitre 5 (cf. page 59) traitant des contraintes souples de seuil. L'encodage booléen des motifs sera utilisé dans chaque chapitre présentant nos contributions.

3.1 Modélisation sous forme d'un CSP

3.1.1 Cadre général

Soit $\mathcal{T}$ l'ensemble des m transactions d'un jeu de données $\boldsymbol{r}$ et $\mathcal{I}$ l'ensemble de ses n items. L'extraction de k motifs satisfaisant un ensemble de contraintes $\mathcal{C}$ peut se modéliser à l'aide d'un CSP (cf. le chapitre 2 à la page 33) $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ où :

— $\mathcal{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$, où chaque variable $\mathtt{x}_i$ $(1 \leq i \leq k)$ représente un motif inconnu,
— $\mathcal{D} = \{D_{\mathtt{x}_1}, \ldots, D_{\mathtt{x}_k}\}$, le domaine initial $D_{\mathtt{x}_i}$ de chaque variable $\mathtt{x}_i$ est $\mathcal{L}_{\mathcal{I}}$.
— $\mathcal{C}$ l'ensemble des contraintes portant sur les k motifs $\mathtt{x}_1$, …, $\mathtt{x}_k$.

Les contraintes sont des relations portant sur des termes, ces termes étant construits en utilisant des constantes, des variables, des opérateurs et des symboles de fonctions désignant des mesures, comme par exemple la taille et la fréquence. $\mathcal{C}$ est généralement composé de contraintes numériques et de contraintes ensemblistes.

3.1.2 Modélisation des règles d'exception

Dans cette section, nous reprenons l'exemple des règles d'exception, et nous montrons comment elles peuvent se modéliser sous forme d'un CSP.

D'après la définition 1.27 (cf. page 23), nous avons :

$$\texttt{exception}(\mathtt{x},\mathtt{y},\mathtt{j}) \equiv \begin{cases} \texttt{freq}((\mathtt{x}\setminus\mathtt{y})\sqcup\mathtt{j}) \geq minfr \wedge \\ \texttt{freq}(\mathtt{x}\setminus\mathtt{y}) - \texttt{freq}((\mathtt{x}\setminus\mathtt{y})\sqcup\mathtt{j}) \leq \delta_1 \\ \wedge \\ \texttt{freq}(\mathtt{x}\sqcup\neg\mathtt{j}) \leq maxfr \wedge \\ \texttt{freq}(\mathtt{x}) - \texttt{freq}(\mathtt{x}\sqcup\neg\mathtt{j}) \leq \delta_2 \end{cases}$$

Que l'on peut reformuler à l'aide de contraintes élémentaires, comme le montre le tableau 3.1 (Pour plus de détails, se reporter à la thèse de Mehdi Khiari [Khiari, 2012]).

Contrainte	Reformulation
$\texttt{freq}((\mathtt{x}\setminus\mathtt{y})\sqcup\mathtt{j}) \geq minfr$	$\texttt{freq}(\mathtt{x}_2) \geq minfr$ $\wedge\ \mathtt{j} \in \mathtt{x}_2$ $\wedge\ \mathtt{x}_1 \subsetneq \mathtt{x}_3$
$\texttt{freq}(\mathtt{x}\setminus\mathtt{y}) - \texttt{freq}((\mathtt{x}\setminus\mathtt{y})\sqcup\mathtt{j}) \leq \delta_1$	$\texttt{freq}(\mathtt{x}_1) - \texttt{freq}(\mathtt{x}_2) \leq \delta_1$ $\wedge\ \mathtt{x}_2 = \mathtt{x}_1 \sqcup \mathtt{j}$
$\texttt{freq}(\mathtt{x}\sqcup\neg\mathtt{j}) \leq maxfr$	$\texttt{freq}(\mathtt{x}_4) \leq maxfr$ $\wedge\ \mathtt{j} \notin \mathtt{x}_4$
$\texttt{freq}(\mathtt{x}) - \texttt{freq}(\mathtt{x}\sqcup\neg\mathtt{j}) \leq \delta_2$	$\texttt{freq}(\mathtt{x}_3) - \texttt{freq}(\mathtt{x}_4) \leq \delta_2$ $\wedge\ \mathtt{x}_4 = \mathtt{x}_3 \sqcup \neg\mathtt{j}$

TABLE 3.1 – Reformulation à l'aide des contraintes élémentaires.

Le CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ modélisant les règles d'exception est alors le suivant :

a) $\mathcal{X} = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{x}_4\}$ est l'ensemble des variables qui représentent les motifs inconnus, avec :
   — pour la règle générale, $\mathtt{x}_1$ associée à $\mathtt{x}\setminus\mathtt{y}$, et $\mathtt{x}_2$ associée à $(\mathtt{x}\setminus\mathtt{y})\sqcup\mathtt{j}$,
   — pour la règle déviationnelle, $\mathtt{x}_3$ associée à $\mathtt{x}$, et $\mathtt{x}_4$ associée à $\mathtt{x}\sqcup\neg\mathtt{j}$.

b) $\mathcal{D} = \{D_{\mathtt{x}_1}, D_{\mathtt{x}_2}, D_{\mathtt{x}_3}, D_{\mathtt{x}_4}\}$ est l'ensemble des domaines où chaque $D_{\mathtt{x}_i} = \mathcal{L}_{\mathcal{I}}$

c) $\mathcal{C} = \mathcal{C}_{ens} \cup \mathcal{C}_{num}$ est l'ensemble des contraintes, avec :
   — le sous-ensemble des contraintes ensemblistes :

$$\mathcal{C}_{ens} = \{(\mathtt{j} \in \mathtt{x}_2), (\mathtt{x}_2 = \mathtt{x}_1 \sqcup \mathtt{j}), (\mathtt{j} \notin \mathtt{x}_4), (\mathtt{x}_4 = \mathtt{x}_3 \sqcup \neg\mathtt{j}), (\mathtt{x}_1 \subsetneq \mathtt{x}_3)\}$$

   — le sous-ensemble des contraintes numériques :

$$\mathcal{C}_{num} = \{(\texttt{freq}(\mathtt{x}_2) \geq minfr), (\texttt{freq}(\mathtt{x}_1) - \texttt{freq}(\mathtt{x}_2) \leq \delta_1), (\texttt{freq}(\mathtt{x}_4) \leq maxfr), (\texttt{freq}(\mathtt{x}_3) - \texttt{freq}(\mathtt{x}_4) \leq \delta_2)\}$$

3.2 Encodage booléen

Cette section présente l'encodage booléen des motifs inconnus recherchés. En particulier, elle montre comment sont établies les relations entre les motifs recherchés, leur support et le jeu de données transactionnel. Tout d'abord, nous présentons l'encodage booléen dans le cas unaire (motifs locaux). Puis, nous nous intéressons au cas n-aire. Nous terminons en développant l'encodage des règles d'exception à partir de la modélisation CSP présentée à la section 3.1.2.

3.2.1 Encodage booléen d'un unique motif

L'encodage booléen d'un unique motif a été introduit par [De Raedt *et al.*, 2008] et utilise des contraintes réifiées (cf. la section 2.4 à la page 39) pour traduire la relations entre ce motif, son support et le jeu de données transactionnel.

Soit $\boldsymbol{r}$ un jeu de données, $\mathcal{T}$ l'ensemble de ses m transactions, et $\mathcal{I}$ l'ensemble de ses n items. On considère la matrice booléenne $\mathbf{d} = (d_{t,i})_{t\in\mathcal{T}, i\in\mathcal{I}}$ associée à $\boldsymbol{r}$ et définie par :

$$\forall t \in \mathcal{T}, \forall i \in \mathcal{I}, (d_{t,i} = 1) \ ssi \ (i \in t)$$

Soit $\mathtt{x}$ le motif unique recherché. $\mathtt{x}$ est représenté par n variables booléennes $\{X_1, X_2, \ldots, X_n\}$ tel que :

$$\forall i \in \mathcal{I}, \ (X_i = 1) \ ssi \ (i \in \mathtt{x})$$

Le support du motif recherché $\mathtt{x}$ est représenté par m variables booléennes $\{T_1, T_2, \ldots, T_m\}$ tel que :

$$\forall t \in \mathcal{T}, \ (T_t = 1) \ ssi \ (\mathtt{x} \subseteq t)$$

La relation entre le motif recherché $\mathtt{x}$, son support et le jeu de données $\boldsymbol{r}$ est établie via des contraintes réifiées (cf. la section 2.4 à la page 39) imposant que :

$$\forall t \in \mathcal{T}, (T_t = 1) \iff (\mathtt{x} \subseteq t) \tag{3.1}$$

ce qui se reformule en [De Raedt *et al.*, 2008] :

$$\forall t \in \mathcal{T}, (T_t = 1) \iff \sum_{i\in\mathcal{I}} X_i \times (1 - d_{t,i}) = 0 \tag{3.2}$$

$\mathcal{T}$	Classe	Items				
t_1	j_1	A		C	D	
t_2	j_1		B	C		E
t_3	j_1	A	B	C		
t_4	j_2		B			E
t_5	j_2	A	B	C		E
t_6	j_2		B	C		E

$\mathcal{T}$ \ $\mathcal{I}$	1	2	3	4	5	6	7
t_1	1	0	1	0	1	1	0
t_2	1	0	0	1	1	0	1
t_3	1	0	1	1	1	0	0
t_4	0	1	0	1	0	0	1
t_5	0	1	1	1	1	0	1
t_6	0	1	0	1	1	0	1

TABLE 3.2 – Jeu de données transactionnel $\boldsymbol{r}$ du tableau 1.1 (cf. page 12)

Exemple 3.1.
Pour le jeu de données transactionnel décrit au tableau 3.2, on obtient l'ensemble suivant de contraintes (les items j_1, j_2, A, B, C, D, E étant numérotés de 1 à 7) :

$$
\begin{array}{l}
(T_1 = 1) \iff (X_2 = 0 \wedge X_4 = 0 \wedge X_7 = 0) \\
(T_2 = 1) \iff (X_2 = 0 \wedge X_3 = 0 \wedge X_6 = 0) \\
(T_3 = 1) \iff (X_2 = 0 \wedge X_6 = 0 \wedge X_7 = 0) \\
(T_4 = 1) \iff (X_1 = 0 \wedge X_3 = 0 \wedge X_5 = 0 \wedge X_6 = 0) \\
(T_5 = 1) \iff (X_1 = 0 \wedge X_6 = 0) \\
(T_6 = 1) \iff (X_1 = 0 \wedge X_3 = 0 \wedge X_6 = 0)
\end{array}
$$

Pour encoder un unique motif inconnu, il faut $(n + m)$ variables booléennes et m contraintes réifiées, une contrainte réifiée étant d'arité au plus $(n + 1)$.

3.2.2 Encodage des contraintes unaires

L'encodage booléen d'un motif permet d'exprimer simplement certaines mesures usuelles :
— la fréquence d'un motif $\mathtt{x}$: $\mathtt{freq}(\mathtt{x}) = \sum_{t \in \mathcal{T}} T_t$
— la taille d'un motif $\mathtt{x}$: $\mathtt{taille}(\mathtt{x}) = \sum_{i \in \mathcal{I}} X_i$

La contrainte de fréquence minimale (par rapport à un seuil $minfr$) se formule par :

$$\sum_{t \in \mathcal{T}} T_t \geq minfr \tag{3.3}$$

celle de taille maximale (par rapport à un seuil Ω) par :

$$\sum_{i \in \mathcal{I}} X_i \leq \Omega \tag{3.4}$$

La contrainte $\mathtt{fermé}(\mathtt{x})$ impose que le motif $\mathtt{x}$ soit fermé par rapport à la fréquence :

$$\forall i \in \mathcal{I},\ (X_i = 1) \iff \sum_{t \in \mathcal{T}} T_t(1 - d_{t,i}) = 0 \tag{3.5}$$

3.2.3 Encodage booléen de plusieurs motifs

Cet encodage proposé par [Khiari *et al.*, 2010b] est une extension du cas unaire présenté à la section 3.2.1. Soient $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_k$ les k motifs recherchés.

Chaque motif inconnu $\mathtt{x}_j$ $(1 \leq j \leq k)$ est modélisé par n variables booléennes $\{X_{1,j}, X_{2,j}, \ldots, X_{n,j}\}$ telles que $(X_{i,j} = 1)$ si et seulement si l'item i appartient au motif $\mathtt{x}_j$:

$$\forall i \in \mathcal{I}, (X_{i,j} = 1) \quad \iff \quad (i \in \mathtt{x}_j)$$

Le support de chaque motif $\mathtt{x}_j$ est représenté par m variables booléennes $\{T_{1,j}, T_{2,j}, \ldots, T_{m,j}\}$ associées à $\mathtt{x}_j$ telles que $(T_{t,j} = 1)$ si et seulement si $(\mathtt{x}_j \subseteq t)$:

$$\forall t \in \mathcal{T}, (T_{t,j} = 1) \quad \iff \quad (\mathtt{x}_j \subseteq t)$$

La relation entre chaque motif recherché $\mathtt{x}_j$ $(1 \leq j \leq k)$, son support et le jeu de données $\boldsymbol{r}$ est établie via des contraintes réifiées imposant que, pour chaque transaction t, $(T_{t,j} = 1)$ si et seulement si $(\mathtt{x}_j \subseteq t)$, ce qui se reformule en :

$$\forall j \in [1..k], \forall t \in \mathcal{T}, (T_{t,j} = 1) \iff \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0 \tag{3.6}$$

3.2.4 Encodage des contraintes n-aires

De manière analogue au cas unaire, cette représentation permet de modéliser simplement certaines mesures usuelles :

— la fréquence d'un motif $\mathtt{x}_j$: $\mathtt{freq}(\mathtt{x}_j) = \sum_{t \in \mathcal{T}} T_{t,j}$,

— la cardinalité d'un motif $\mathtt{x}_j$: $\mathtt{taille}(\mathtt{x}_j) = \sum_{i \in \mathcal{I}} X_{i,j}$.

La contrainte de fréquence minimale (par rapport à un seuil $minfr_j$) se formule par :

$$\sum_{t \in \mathcal{T}} T_{t,j} \geq minfr_j \tag{3.7}$$

celle de taille maximale (par rapport à un seuil Ω_j) par :

$$\sum_{i \in \mathcal{I}} X_{i,j} \leq \Omega_j \tag{3.8}$$

La contrainte $\mathtt{fermé}(\mathtt{x}_j)$ impose que le motif $\mathtt{x}_j$ soit fermé par rapport à la fréquence :

$$\forall i \in \mathcal{I}, (X_{i,j} = 1) \iff \sum_{t \in \mathcal{T}} T_{t,j}(1 - d_{t,i}) = 0 \tag{3.9}$$

A titre d'exemple, le tableau 3.3 décrit l'encodage booléen de quelques contraintes ensemblistes.

Contrainte	**Encodage booléen**
$X_p = X_q$	$\forall i \in \mathcal{I}, X_{i,p} = X_{i,q}$
$X_p \subseteq X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q}$
$X_p \subset X_q$	$\forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q} \ \wedge$ $\sum_{i \in \mathcal{I}} X_{i,p} < \sum_{i \in \mathcal{I}} X_{i,q}$
$i_o \in X_p$	$X_{i_o,p} = 1$
$X_p \cap X_q = \emptyset$	$\forall i \in \mathcal{I}, X_{i,p} + X_{i,q} \leq 1$

TABLE 3.3 – Exemples d'encodage de contraintes ensemblistes.

L'encodage booléen d'une requête portant sur k motifs inconnus, requiert :

— $(k \times (n+m))$ variables booléennes représentant respectivement les motifs inconnus et leur support.

— $(k \times m)$ contraintes réifiées, une contrainte réifiée étant d'arité au plus $(n+1)$.

3.2.5 Exemple des règles d'exception

Considérons le jeu de données $\boldsymbol{r}$ du tableau 3.2 (cf. page 47), et réalisons l'encodage booléen des règles d'exception à partir de la modélisation présentée à la section 3.1.2 (cf. page 46). Nous avons $(k = 4)$ motifs inconnus, $(n = 7)$ items et $(m = 6)$ transactions.

A chaque motif inconnu $\mathtt{x}_j$ ($j \in [1..k]$), sont associées :
- ($n = 7$) variables booléennes $X_{1,j}, X_{2,j}, \ldots, X_{7,j}$ représentant le motif $\mathtt{x}_j$,
- ($m = 6$) variables booléennes $T_{1,j}, T_{2,j}, \ldots, T_{6,j}$ représentant le support de $\mathtt{x}_j$,

Pour chaque motif inconnu $\mathtt{x}_j$ ($j \in [1..k]$), il faut imposer les m contraintes réifiées suivantes :

$$\begin{array}{l}
(T_{1,j} = 1) \iff (X_{2,j} = 0 \wedge X_{4,j} = 0 \wedge X_{7,j} = 0) \\
(T_{2,j} = 1) \iff (X_{2,j} = 0 \wedge X_{3,j} = 0 \wedge X_{6,j} = 0) \\
(T_{3,j} = 1) \iff (X_{2,j} = 0 \wedge X_{6,j} = 0 \wedge X_{7,j} = 0) \\
(T_{4,j} = 1) \iff (X_{1,j} = 0 \wedge X_{3,j} = 0 \wedge X_{5,j} = 0 \wedge X_{6,j} = 0) \\
(T_{5,j} = 1) \iff (X_{1,j} = 0 \wedge X_{6,j} = 0) \\
(T_{6,j} = 1) \iff (X_{1,j} = 0 \wedge X_{3,j} = 0 \wedge X_{6,j} = 0)
\end{array}$$

A ces contraintes réifiées, il faut ajouter les contraintes issues de l'encodage booléen des contraintes élémentaires associées aux règles d'exception. Ces contraintes élémentaires sont décrites colonne de droite du tableau 3.1. Le tableau 3.4 ci-dessous décrit cet encodage.

Contrainte	**Encodage booléen**
$\mathtt{x}_1 \subset \mathtt{x}_3$	$\forall i \in \mathcal{I},\ X_{i,1} \leq X_{i,3} \wedge \sum_{i \in \mathcal{I}} X_{i,1} < \sum_{i \in \mathcal{I}} X_{i,3}$
$\mathtt{x}_2 = \mathtt{x}_1 \sqcup \mathsf{j}$	$\forall i \in \mathcal{I} \setminus \{5\}, X_{i,2} = X_{i,1} \wedge X_{5,2} = 1 \wedge X_{5,1} = 0$
$\mathtt{x}_4 = \mathtt{x}_3 \sqcup \neg\mathsf{j}$	$\forall i \in \mathcal{I} \setminus \{6\}, X_{i,4} = X_{i,3} \wedge X_{6,4} = 1 \wedge X_{6,3} = 0$
$\mathtt{freq}(\mathtt{x}_2) \geq minfr$	$\sum_{t \in \mathcal{T}} T_{t,2} \geq minfr$
$\mathtt{freq}(\mathtt{x}_1) - \mathtt{freq}(\mathtt{x}_2) \leq \delta_1$	$\sum_{t \in \mathcal{T}} (T_{t,1} - T_{t,2}) \leq \delta_1$
$\mathtt{freq}(\mathtt{x}_4) \leq maxfr$	$\sum_{t \in \mathcal{T}} T_{t,4} \leq maxfr$
$\mathtt{freq}(\mathtt{x}_3) - \mathtt{freq}(\mathtt{x}_4) \leq \delta_2$	$\sum_{t \in \mathcal{T}} (T_{t,3} - T_{t,4}) \leq \delta_2$

TABLE 3.4 – Encodage booléen des contraintes élémentaires (règles d'exception).

3.3 Conclusion

Dans ce chapitre, nous avons présenté la modélisation des problèmes d'extraction de motifs sous forme de CSP, ainsi que l'encodage booléen des motifs recherchés. La modélisation d'une requête sous forme d'un CSP sera plus particulièrement utilisée au chapitre 5 (cf. page 59) traitant des contraintes souples de seuil. L'encodage booléen des motifs sera utilisé par chacune de nos contributions.

"Ce que je préfère de New York est la vue, le skyline."

Jack Reynor

Chapitre 4

Skylines

Sommaire

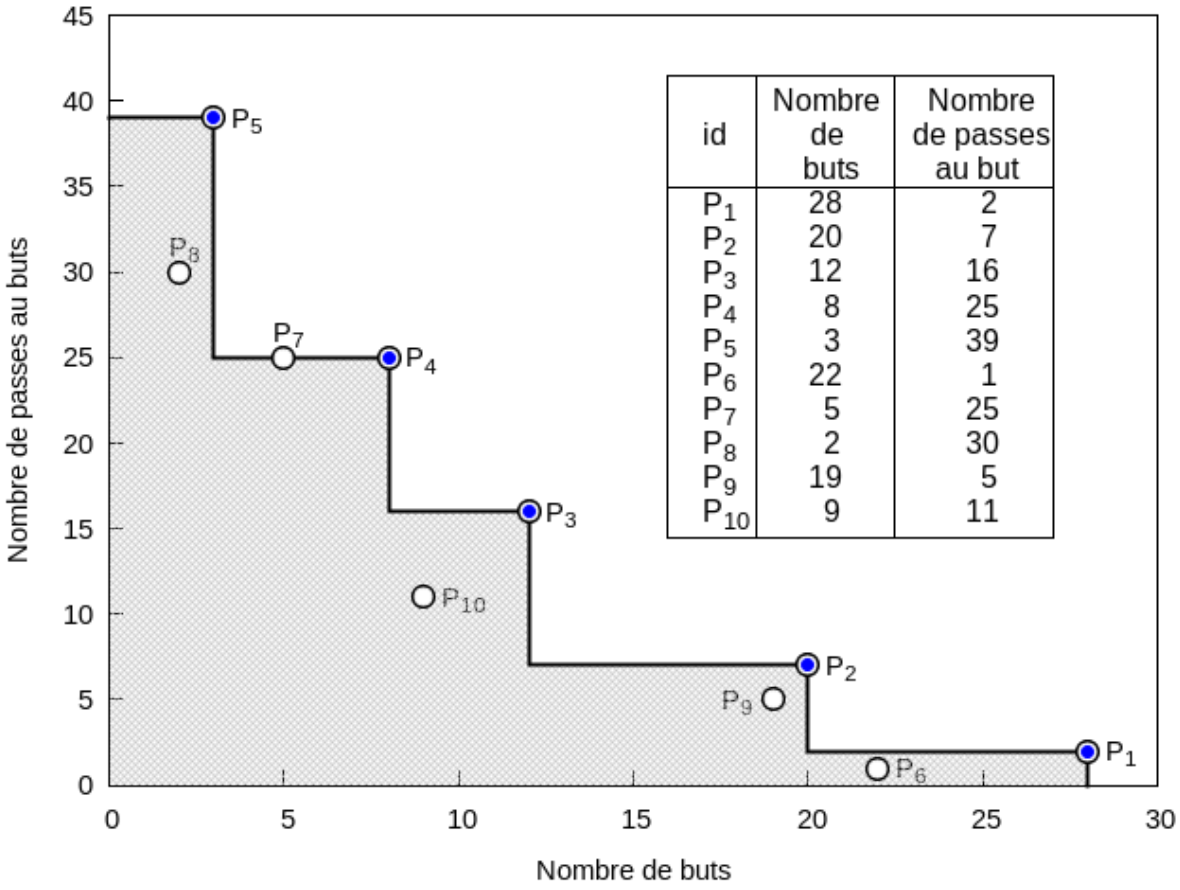
Ce chapitre introduit la problématique de l'extraction des points skylines dans les bases de données multidimensionnelles. Tout d'abord, nous introduisons les requêtes skylines (cf. la section 4.1). Puis, nous nous intéressons aux algorithmes de calcul des skylines dans le contexte des bases de données multidimensionnelles (cf. la section 4.2). En section 4.3, nous présentons les rares travaux existants sur la relaxation de skylines. Enfin, nous présentons en section 4.4 le concept de cube de skylines (encore appelée Skycube), et nous discutons les travaux existants sur le calcul du Skycube.

4.1 Requêtes skyline

Les requêtes skylines permettent d'exprimer les préférences de l'utilisateur à partir d'une relation de dominance. Elles ont été largement étudiées dans les communautés des bases de données et de l'intelligence artificielle. Dans un espace multidimensionnel, où une préférence est définie pour chaque dimension, les requêtes skylines déterminent les points qui sont meilleurs ou égaux sur toutes les dimensions et strictement meilleurs sur au moins une dimension.

4.1.1 Exemple introductif

Considérons un entraîneur d'une équipe de football qui souhaite recruter des joueurs pour la saison prochaine (cf. la figure 4.1). Chaque joueur est représenté selon les dimensions suivantes : le nombre



id	Nombre de buts	Nombre de passes au but
P_1	28	2
P_2	20	7
P_3	12	16
P_4	8	25
P_5	3	39
P_6	22	1
P_7	5	25
P_8	2	30
P_9	19	5
P_{10}	9	11

FIGURE 4.1 – Exemple illustratif.

de buts marqués et le nombre de passes décisives effectuées au cours de la dernière saison. Nous allons illustrer les définitions de dominance, de skylines et de relation de préférence à l'aide de cet exemple.

4.1.2 Définitions

Soient $D = \{d_1, \ldots, d_l\}$ un espace à l dimensions, E un ensemble de points définis dans l'espace D et $p \in E$ un point, on notera par x_i^p la valeur de p sur la dimension d_i.

Sur l'exemple introductif de la section 4.1.1, l'ensemble E contient 10 points décrits par deux dimensions : (d_1) le nombre de buts marqués et (d_2) le nombre de passes décisives effectuées. Les valeurs selon chaque dimension sont totalement ordonnées par la relation d'ordre $\geq$. Elle spécifie qu'on préfère les joueurs ayant un nombre de buts marqués (respectivement le nombre de passes décisives effectuées) le plus élevé.

Définition 4.1. PARETO-DOMINANCE
Étant donné un espace $D = \{d_1, \ldots, d_l\}$ à l dimensions et un ensemble de points E, un point $p \in E$

domine un autre point $q \in E$ selon D (dénoté par $p \succ_D q$), si et seulement si, $\forall d_i \in D, x_i^p \geq x_i^q$ et $\exists d_j \in D, x_j^p > x_j^q$.

Définition 4.2. Skyline
L'ensemble des skylines de E selon D est l'ensemble des points de E qui ne sont dominés (selon D) par aucun autre point de E :

$$Skyline(D, E) = \{p \in E \mid \nexists\, q \in E, q \succ_D p\}$$

Reprenons l'exemple introductif de la section 4.1.1, $Skyline(D, E) = \{p_1, p_2, p_3, p_4, p_5\}$. En effet, les joueurs p_6, p_7, p_8, p_9 et p_{10} sont dominés par au moins un autre joueur, donc ils ne peuvent pas être des skylines. De plus, les joueurs p_1, p_2, p_3, p_4 et p_5 ne sont pas dominés par aucun autre joueur. La figure 4.1 représente graphiquement ces skylines.

4.2 Calcul des skylines

La notion de dominance que nous avons présentée dans la section 4.1.2 est au cœur du calcul des skylines. Cette tâche est une dérivation du problème de vecteur maximal en géométrie computationnelle [Matousek, 1991], la frontière de Pareto [Kung *et al.*, 1975] et l'optimisation multi-critères [Steuer, 1992]. Depuis sa redécouverte au sein de la communauté de base de données par [Börzsönyi *et al.*, 2001], divers travaux de recherche se sont intéressés au calcul des skylines en présence de larges volumes de données [Börzsönyi *et al.*, 2001, Tan *et al.*, 2001, Chomicki *et al.*, 2005, Papadias *et al.*, 2008]. Ces travaux portent sur l'optimisation du calcul des skylines associés à toutes les dimensions de l'espace considéré.

Dans cette section, nous présentons les principaux algorithmes de calcul de skylines. Nous distinguons deux familles d'algorithmes selon qu'ils utilisent ou non des techniques d'indexation. Les premiers n'utilisent pas de structure de données particulière et la recherche des skylines se fait en analysant entièrement la base de données au moins une fois. Les seconds regroupent les approches qui exploitent des index afin d'améliorer l'efficacité de calcul et d'accélérer les requêtes skylines, en diminuant par exemple le nombre de tests de dominance.

4.2.1 Les algorithmes sans index

A) L'algorithme à boucles imbriqués (Block Nested Loop BNL) Il s'agit du premier algorithme de recherche de skylines qui a été proposé dans le contexte de bases de données [Börzsönyi *et al.*, 2001]. L'idée principale de cet algorithme est de comparer tous les points de l'ensemble de données deux à deux (i.e. tests de dominance), tout en gardant une liste de points candidats au skyline en mémoire centrale. En cas de saturation de la mémoire centrale, l'algorithme fait appel à un fichier temporaire stocké en mémoire secondaire dans lequel sont stockés tous les candidats non considérés. Il seront traités lors d'une prochaine itération. Lors de la comparaison d'un point p avec les points stockés en mémoire centrale, trois cas de figure se présentent :

- *le point p est dominé par un point présent en mémoire centrale* : p est alors directement éliminé de l'ensemble skyline (i.e. il est dominé) et ne sera plus pris en compte pour le reste des calculs.
- *le point p domine un ou plusieurs points présents en mémoire centrale* : les points dominés par p sont directement éliminés de la liste de points candidats au skyline. Le point p quant à lui est inséré en mémoire centrale.

— *le point p est incomparable* [14] avec l'ensemble des points présents en mémoire centrale : dans ce cas, si la mémoire n'est pas pleine, p est alors inséré en mémoire centrale. Sinon, il est mis de côté dans le fichier temporaire cité précédemment et sera examiné à nouveau au cours de la prochaine itération de l'algorithme.

La complexité de l'algorithme dans le pire cas est quadratique $\mathcal{O}(n^2)$ où n représente le cardinal de l'ensemble de données E.

B) L'algorithme Divide & Conquer (D&C) a été proposé par Börzsönyi *et al.* [Börzsönyi *et al.*, 2001]. Le principe général de cet algorithme consiste à travailler de manière récursive en divisant l'ensemble de données en plusieurs partitions de sorte à ce que chaque partition puisse tenir en mémoire centrale. Le skyline partiel des points de chaque partition est calculé en utilisant un algorithme de recherche de skylines se basant sur un calcul en mémoire centrale (ex. l'algorithme BNL). Le skyline final est ensuite obtenu en fusionnant tous les skylines partiels. La complexité de cet algorithme est de l'ordre de $\mathcal{O}(n.(\log n)^{l-2} + n.\log n)$ où n représente le cardinal de l'ensemble de données E, et l le nombre de dimensions.

C) L'algorithme Sort Filter Skyline (SFS) [Chomicki *et al.*, 2003, Chomicki *et al.*, 2005] est une version améliorée de l'algorithme BNL en triant d'abord les points en entrée à l'aide d'une fonction de score *monotone* $\mathfrak{f}$. Cela garantit que si $\mathfrak{f}(p) \leq \mathfrak{f}(q)$, alors q ne dominera jamais p. En d'autres termes, la fonction monotone correspond à l'exécution d'un tri topologique par rapport au critère de la dominance de Pareto. Ainsi, comme pour BNL, SFS garde en mémoire centrale tous les points non dominés vues jusqu'ici. Toutefois, la monotonie de la fonction $\mathfrak{f}$ garantit que tout nouveau point p inséré en mémoire centrale (i.e. p est incomparable avec l'ensemble des points déjà présents en mémoire) ne peut être dominé par n'importe quel autre point q de l'ensemble de données, réduisant ainsi le nombre de tests de dominance. De plus, l'algorithme retourne les points skylines au fur et à mesure de leurs calculs (i.e. algorithme dit progressif). La complexité de l'algorithme dans le pire cas est quadratique $\mathcal{O}(n^2)$ (i.e. la même que celle du BNL) où n représente le cardinal de l'ensemble de données E, car avant d'effectuer le test de dominance, il fait un tirage de points.

4.2.2 Les algorithmes avec index

A) L'algorithme Index [Tan *et al.*, 2001] organise l'ensemble des données en l listes de sorte qu'un point $p = (x_1^p, x_2^p, \ldots, x_l^p)$ est affecté à la $i$ème liste ($1 \leq i \leq l$), si et seulement si, sa coordonnée x_i^p sur la dimension i correspond à la plus petite valeur de p sur toutes les dimensions. Les points de chaque liste sont triés dans l'ordre croissant de leur coordonnée minimale et indexés sous forme d'un arbre B-Tree. Ensuite, l'algorithme analyse les l listes de manière séquentielle et simultanée à partir des groupes de points [15] de chaque liste. Il s'agit de calculer les skylines locaux de chaque groupe de points en utilisant les B-Tree, puis de fusionner ces derniers avec les autres skylines pour former le skyline final.

B) L'algorithme Nearest Neighbor (NN) [Kossmann *et al.*, 2002] est fondé sur des requêtes de plus proche voisin qui sont implémentées à l'aide d'index de type R/R*-Tree. NN utilise une constatation géométrique très simple : les points les plus intéressants sont ceux qui sont proches de l'origine du repère. L'algorithme, commence en cherchant le point ayant la plus petite distance de l'origine du repère. Ce dernier représentant forcément un point skyline p. Ensuite, il segmente le reste des points en l partitions (i.e. zones de l'espace), une partition étant ajoutée pour chaque coordonnée de p. Les prochains plus proches voisins sont ensuite itérativement trouvés dans chaque partition (régions susceptibles de contenir d'autres points candidats).

14. Deux points p et q ($p \neq_D q$) sont *incomparables* par rapport à D (dénoté $p \prec\succ_D q$) ssi $p \not\succ_D q$ et $q \not\succ_D p$.

15. Un groupe de points pour chaque liste, représente l'ensemble des points ayant la même valeur sur la dimension correspondante.

4.2.3 Approche PPC pour le calcul de la frontière Pareto

[Gavanelli, 2002] a proposé un algorithme pour déterminer la frontière Pareto en utilisant une approche CSP. Cet algorithme est fondé sur le concept de nogoods [16] et utilise des structures de données spatiales (quadtrees) pour organiser l'ensemble des nogoods. L'approche proposée ne traite que les points non-dominés (durs). En outre, elle ne peut pas être appliquée pour le calcul des skypatterns.

4.3 Introduire de la souplesse pour les requêtes skylines

Comme nous l'avons vu précédemment, les requêtes skylines permettent d'exprimer les souhaits de l'utilisateur selon la relation de *dominance*. Toutefois, dans certaines situations, les résultats obtenus ne sont pas toujours entièrement satisfaisants pour l'utilisateur. En effet, considérons à nouveau notre exemple introductif de la section 4.1.1. L'entraîneur sera intéressé par des joueurs non-skylines s'il cherche :

- les joueurs en position d'attaquant : l'entraîneur donnera la priorité au nombre de buts marqués. Les joueurs p_6 (non-skyline) et p_9 (non-skyline) deviennent intéressants.
- les joueurs évoluant à la position de milieu de terrain offensif : l'entraîneur donnera la priorité au nombre de passes décisives effectuées. Les joueurs p_7 (non-skyline) et p_8 (non-skyline) deviennent intéressants.
- les joueurs polyvalents : l'entraîneur donnera la priorité à l'équilibre entre le nombre de buts marqués et le nombre de passes décisives effectuées. Le joueur p_{10} (non-skyline) devient intéressant.

De plus, les joueurs skylines sont très demandés et onéreux : ils pourraient être embauchés par une autre équipe ou leurs salaires pourraient être hors de budget (i.e. faramineux). Ainsi, les joueurs non-skylines, qui sont proches de joueurs skylines, peuvent être d'un grand intérêt pour l'entraîneur. Ces joueurs peuvent être découverts en relâchant légèrement la relation de dominance.

L'idée d'introduire de la souplesse dans les requêtes skylines a été (très) peu abordée dans la littérature de la communauté de base de données. À notre connaissance, seuls les *thick skylines*, proposés par [Jin *et al.*, 2004], permettent d'étendre le concept de skyline aux voisins proches. Un thick skyline est soit un point skyline p_i, ou un point p_j dominé par un point skyline p_i, tel que p_j est situé dans le voisinage de p_i, défini par la demi-cercle de rayon $\epsilon > 0$ et de centre p_i.

4.4 Cubes de skylines

Dans la pratique, les utilisateurs ne connaissent pas le rôle exact de chaque dimension et préfèrent poser des requêtes sur différents sous-ensembles de dimensions et non sur la totalité. Pour répondre à ce type de problématique, divers travaux se sont intéressés à l'extraction des points skylines dans des sous-espaces de dimensions. [Pei *et al.*, 2005, Pei *et al.*, 2006, Yuan *et al.*, 2005] sont les premiers à proposer une généralisation *multidimensionnelle* du skyline à travers le concept de Skycube. Cette structure réunit l'ensemble de tous les skylines dans tous les sous-espaces non vides possibles (appelés aussi *cuboïdes*). Il est alors possible de rechercher efficacement des points dominants selon différentes combinaisons de critères. De plus, grâce à cette structure, il devient possible d'observer le comportement des points skylines à travers l'espace multidimensionnel et ainsi d'analyser et de comprendre les différents facteurs de dominance. Par exemple, les nouveaux skylines qui apparaissent et ceux qui disparaissent par l'ajout et/ou suppression d'une dimension.

16. Un nogood est une affectation partielle ou complète des variables permettant d'éviter des explorations redondantes de l'espace de recherche.

Le SKYCUBE, étant inspiré du *cube de données* [Gray *et al.*, 1997], souffre des mêmes inconvénients de coût de calcul et d'explosion de l'espace de stockage. Ainsi il est naturel d'essayer d'en proposer des représentations concises et des algorithmes efficaces associés. Nous dressons ci-dessous un bref état de l'art des algorithmes de calcul du SKYCUBE.

Dans [Yuan *et al.*, 2005] deux approches ont étés proposées :
- Bottom-Up Skycube (BUS) : L'idée de base de l'algorithme BUS est de calculer chaque cuboïde du SKYCUBE niveau par niveau et de bas en haut, où les cuboïdes fils sont fusionnés pour former (en partie) les cuboïdes pères de niveau supérieur.
- Top-Down Skycube (TDS) : À l'inverse de BUS, l'algorithme TDS permet de calculer chaque cuboïde du SkyCube niveau par niveau et de haut en bas en étendant le principe de l'algorithme D&C.

Pei et *al.* [Pei *et al.*, 2007] ont proposé STELLAR, une méthode de calcul du SKYCUBE qui évite de rechercher l'ensemble des skylines pour chaque cuboïde. En utilisant les notions de Groupe Skyline et de Sous-espace décisif, STELLAR garantit qu'en conservant uniquement les groupes skylines et les sous-espaces décisifs associés, il est possible de retrouver tous les points skylines. Cette représentation est donc plus concise que le SKYCUBE.

L'algorithme ORION [Raïssi *et al.*, 2010] est à notre connaissance l'approche la plus récente pour le calcul et l'optimisation du SKYCUBE. Il propose une représentation concise du SKYCUBE. Il réduit considérablement les tests de dominance dans un sous-espace donné en identifiant les points skylines qui peuvent être dérivés à partir des skylines d'autres sous-espaces. En se basant sur la connexion de Galois, l'algorithme réduit également le nombre de sous-espaces à explorer. En effet, un opérateur de fermeture a été défini pour construire une représentation concise du SKYCUBE.

4.5 Conclusion

Dans la première partie de ce chapitre, nous avons motivé et présenté les requêtes skylines, permettant d'exprimer la préférence de l'utilisateur selon la relation de dominance dans les bases de données multidimensionnelles. Nous avons détaillé les différents algorithmes (avec et sans index) de calculs des skylines. Dans la seconde partie, nous avons présenté le concept de SKYCUBE qui représente le cube de tous les skylines de tous les sous-espaces possibles non vides ainsi qu'un bref état de l'art des algorithmes de calcul de SKYCUBE.

Dans la suite de ce manuscrit, nous définirons la notion de skypattern souple, permettant de relaxer la relation de dominance. Nous montrerons comment l'extraction des skypatterns et des skypatterns souples peut être modélisée et résolue avec une approche PPC. Puis, nous introduirons le cube de skypatterns ainsi que deux approches permettant de calculer efficacement ce cube dont l'une utilise la notion de skypattern souple.

Deuxième partie

Contributions

"J'ai eu mes résultats depuis longtemps mais je ne sais pas encore comment je dois arriver à eux."

Carl Friedrich Gauss

"Il fallait, comme pour beaucoup de choses de ce monde, se contenter de rester sur le seuil."

Gustave Flaubert

Chapitre 5

Extraction de motifs sous contraintes souples de seuil

Sommaire

Notre motivation pour l'introduction de la souplesse dans le processus d'extraction de motifs est d'attaquer la *rigidité du cadre actuel*[17] et le *problème de seuillage*[18]. L'idée sous-jacente est d'accepter, comme solutions, les motifs qui ne satisfont pas la requête, mais qui sont "proches". Pour cela, nous introduisons la notion de *contrainte souple de seuil* permettant de relaxer des contraintes portant sur des mesures quelconques. Le cadre que nous avons retenu, pour modéliser l'extraction de motifs sous contraintes souples de seuil, est le modèle disjonctif proposé par Thierry Petit [Petit, 2002] (cf. la section 2.5.3 pour les raisons qui ont motivé notre choix). Pour intégrer la souplesse dans le processus d'extraction des top-k, nous définissons la notion de *top-k motifs souples*. Notre approche bénéficie pleinement de la manipulation des contraintes souples de seuil.

Ce chapitre est organisé comme suit. La section 5.1 montre comment les contraintes souples de seuil peuvent être mises en œuvre dans un extracteur de motifs (fondé sur les CSPs) en utilisant les travaux existants en PPC sur la relaxation de contraintes. La section 5.2 introduit la notion de top-k motif souple. La section 5.3 dresse un état de l'art des rares travaux portant sur la relaxation de contraintes en fouille de données. La section 5.4 décrit en détail la mise en œuvre de notre relaxation sur une application réelle dans le domaine de la chémoinformatique. Les sections 5.5 et 5.6 sont consacrées aux expérimentations : la première porte sur l'extraction des règles d'exception et des règles inattendues sur les jeux de données de l'*UCI*, alors que la seconde est consacrée à notre étude de cas portant sur la découverte de fragments moléculaires toxicophores.

5.1 Contraintes souples de seuil

Cette section présente la première contribution de ce chapitre. Elle décrit comment les contraintes souples de seuil peuvent être mises en œuvre dans un extracteur de motifs (fondé sur les CSPs) en utilisant le modèle disjonctif [Régin *et al.*, 2000] pour la relaxation de contraintes.

Soient $\mathtt{x}$ un motif local, $\{\mathtt{m}_i\}$ un ensemble de mesures, $\{\psi_i\}$ (respectivement $\{\Psi_i\}$) un ensemble des seuils minimaux (respectivement maximaux) correspondant aux mesures $\{\mathtt{m}_i\}$ et une requête $\mathtt{q}(\mathtt{x})$ exprimée sous la forme d'une conjonction de contraintes de la forme $\mathtt{c}_i(\mathtt{x}) \equiv \mathtt{m}_i(\mathtt{x}) \geq \psi_i$ (ou $\mathtt{c}_i(\mathtt{x}) \equiv \mathtt{m}_i(\mathtt{x}) \leq \Psi_i$), telle que $\mathtt{c}_i(\mathtt{x})$ est soit une contrainte dure ou une contrainte souple.

5.1.1 Motivation à l'aide d'un exemple

Considérant le jeu de données transactionnel $\boldsymbol{r}$ du tableau 5.1 où chaque transaction t_i regroupe des articles décrits par des items notés $A, \ldots, F$. L'exemple classique est un jeu de données de supermarché dans lequel chaque transaction correspond à un client et chaque élément de la transaction à un produit acheté par le client. Un prix est associé à chaque produit.

Soit la requête suivante $\mathtt{q}(\mathtt{x})$ permettant d'extraire de la base $\boldsymbol{r}$, tous les motifs fréquents ($minfr = 4$), de taille au moins 3 ($\omega = 3$) et dont la moyenne des prix des articles le composant est supérieure ou égale à 45 ($\psi_{prix} = 45$) :

$$\mathtt{q}(\mathtt{x}) \equiv \mathtt{freq}(\mathtt{x}) \geq 4 \wedge \mathtt{taille}(\mathtt{x}) \geq 3 \wedge \mathtt{moyenne}(\mathtt{x}.prix) \geq 45$$

Par la suite, nous adoptons la notation $\mathtt{x}'$ $\langle v_1, v_2, v_3 \rangle$, où $\mathtt{x}'$ est une solution à la requête $\mathtt{q}(\mathtt{x})$ et v_1, v_2, v_3 désignent les valeurs de ces trois mesures pour $\mathtt{x}'$.

17. Rejet d'un motif, potentiellement intéressant, violant légèrement une ou plusieurs contraintes de la requête.
18. Comment fixer les valeurs de seuils liés aux contraintes ?

Trans.	Items					
t_1		B			E	F
t_2		B	C	D		
t_3	A				E	F
t_4	A	B	C	D	E	
t_5		B	C	D	E	
t_6		B	C	D	E	F
t_7	A	B	C	D	E	F

Item	Prix
A	30
B	40
C	10
D	40
E	70
F	55

TABLE 5.1 – Jeu de données transactionnel $\boldsymbol{r}$.

Sur cet exemple, en considérant uniquement la contrainte de fréquence, il y a 17 solutions. Avec la conjonction des trois contraintes, il subsiste une unique solution : BDE $\langle 4, 3, 50\rangle$. Considérons les 4 motifs suivants qui ne sont pas extraits par la requête $\mathtt{q}(\mathtt{x})$:

- BEF $\langle 3, 3, 55\rangle$
- CDE $\langle 4, 3, 40\rangle$
- BCE $\langle 4, 3, 40\rangle$
- $BCDE$ $\langle 4, 4, 40\rangle$

Le motif BEF satisfait deux des trois contraintes de la requête $\mathtt{q}(\mathtt{x})$, mais viole légèrement la contrainte de fréquence. Toutefois, ce motif est clairement intéressant car il présente une dépense moyenne plus élevée que celle du motif BDE, qui satisfait la requête. Ainsi, en relâchant très légèrement le seuil de fréquence ($\mathtt{freq}(\mathtt{x}) \geq 3$), BEF serait une solution.

De même, relâcher le seuil de la contrainte de prix moyen (de 45 à 40) permettrait d'extraire trois nouveaux motifs : CDE, BCE et $BCDE$. De plus, il est difficile de dire que ces motifs sont moins intéressants que BDE compte-tenu de l'incertitude quant à la valeur fixée du seuil.

Ainsi, la rigidité du cadre de satisfaction fait qu'un motif potentiellement intéressant n'est pas retenu dès qu'une contrainte de seuil est légèrement violée. En outre, il n'est pas raisonnable, dans des applications réelles, de considérer que les contraintes sont toutes d'une importance égale. D'où l'idée d'introduire une certaine souplesse, en relâchant les seuils, pour éviter une sélection dichotomique des motifs.

5.1.2 Mesures de violation pour les contraintes souples de seuil

Lorsqu'on relaxe une contrainte de seuil, nous devons mesurer sa violation. Pour cela, une sémantique de violation μ est associée à la contrainte (cf. la définition 2.8 à la page 40), permettant de quantifier son degré de violation.

Dans cette section, nous proposons plusieurs sémantiques de violation permettant de quantifier la violation sur des contraintes [19] portant sur des mesures. Ces mesures peuvent être quelconques. Dans ce qui suit, nous noterons par $\mathtt{m}$ une mesure quelconque et par $\psi_{\mathtt{m}}$ (resp. $\Psi_{\mathtt{m}}$) un seuil minimal (resp. maximal) pour cette mesure. Soit $\max_{\mathtt{m}}$ la valeur maximale pour la mesure $\mathtt{m}$ [20] et $\mathtt{c}(\mathtt{x})$ la contrainte portant sur $\mathtt{m}$, définie soit par $\mathtt{c}(\mathtt{x}) \equiv \mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}}$ ou par $\mathtt{c}(\mathtt{x}) \equiv \mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}}$.

19. Ces contraintes pouvant être dures ou souples. La relaxation d'une contrainte dure de seuil est elle même.
20. Pour $\mathtt{m} = \mathtt{freq}$, $\max_{\mathtt{m}} = |\mathcal{T}|$; pour $\mathtt{m} = \mathtt{taille}$, $\max_{\mathtt{m}} = |\mathcal{I}|$, etc.

5.1.2a Sémantique de violation μ_1

Une **première sémantique de violation** μ_1 pour la contrainte $\mathtt{c(x)}$ consiste à associer, à chaque motif $\mathtt{x}$, l'écart absolu de $\mathtt{m(x)}$ au seuil (minimal $\psi_\mathtt{m}$ ou maximal $\Psi_\mathtt{m}$) :

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \geq \psi_\mathtt{m} \qquad \rightarrow \mu_1(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \geq \psi_\mathtt{m} \\ \psi_\mathtt{m} - \mathtt{m(x)} & sinon \end{cases}$$

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \leq \Psi_\mathtt{m} \qquad \rightarrow \mu_1(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \leq \Psi_\mathtt{m} \\ \mathtt{m(x)} - \Psi_\mathtt{m} & sinon \end{cases}$$

5.1.2b Sémantique de violation μ_2

Pour pouvoir cumuler les violations de plusieurs contraintes de seuil dans une même requête, il est plutôt souhaitable de travailler avec des écarts relatifs dû à l'hétérogénéité des mesures. Une **seconde sémantique de violation** μ_2 pour la contrainte $\mathtt{c(x)}$ consiste à associer, à chaque motif $\mathtt{x}$, l'écart relatif de $\mathtt{m(x)}$ au seuil (minimal $\psi_\mathtt{m}$ ou maximal $\Psi_\mathtt{m}$) :

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \geq \psi_\mathtt{m} \qquad \rightarrow \mu_2(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \geq \psi_\mathtt{m} \\ \frac{\psi_\mathtt{m} - \mathtt{m(x)}}{\psi_\mathtt{m}} & sinon \end{cases}$$

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \leq \Psi_\mathtt{m} \qquad \rightarrow \mu_2(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \leq \Psi_\mathtt{m} \\ \frac{\mathtt{m(x)} - \Psi_\mathtt{m}}{max_\mathtt{m} - \Psi_\mathtt{m}} & sinon \end{cases}$$

Ainsi, les valeurs de violation pour μ_2 seront de nombres réels dans l'intervalle [0..1].

5.1.2c Sémantique de violation μ_3

Une **troisième sémantique de violation** μ_3 pour la contrainte $\mathtt{c(x)}$ est définie comme suit : si la valeur $\mathtt{m(x)}$ d'un motif $\mathtt{x}$ est jugée trop loin du seuil (minimal $\psi_\mathtt{m}$ ou maximal $\Psi_\mathtt{m}$), alors on considère que la contrainte est insatisfaite. On s'autorise à relaxer dans une proportion de ($\epsilon \times \psi_\mathtt{m}$ ou $\epsilon \times \Psi_\mathtt{m}$), où ϵ est un pourcentage ($\epsilon \in [0..1]$) :

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \geq \psi_\mathtt{m} \qquad \rightarrow \mu_3(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \geq \psi_\mathtt{m} \\ \frac{\psi_\mathtt{m} - \mathtt{m(x)}}{\psi_\mathtt{m}} & si \;\; (1-\epsilon) \times \psi_\mathtt{m} \leq \mathtt{m(x)} \leq \psi_\mathtt{m} \\ \infty & sinon \end{cases}$$

$$\rightarrow \mathtt{c(x)} \equiv \mathtt{m(x)} \leq \Psi_\mathtt{m} \qquad \rightarrow \mu_3(\mathtt{x}) = \begin{cases} 0 & si \;\; \mathtt{m(x)} \leq \Psi_\mathtt{m} \\ \frac{\mathtt{m(x)} - \Psi_\mathtt{m}}{\Psi_\mathtt{m}} & si \;\; \Psi_\mathtt{m} \leq \mathtt{m(x)} \leq (1+\epsilon) \times \Psi_\mathtt{m} \\ \infty & sinon \end{cases}$$

5.1.3 Extraction de motifs sous contraintes souples de seuil

Cette section montre comment les contraintes souples de seuil peuvent être transformées en des contraintes dures équivalentes qui peuvent être traitées directement par un solveur CSP. Tout d'abord, nous commençons par définir formellement le problème d'extraction de motifs sous contraintes souples de seuil. Ensuite, nous détaillons les principales étapes de notre approche.

Définition 5.1. Problème d'extraction de motifs sous contraintes souples de seuil Étant donné un jeu de données $\boldsymbol{r}$, une requête souple $\mathtt{q}(\mathtt{x}) = \bigwedge_{i=1}^{n} \mathtt{c}_i(\mathtt{x})$, un seuil de violation maximal autorisé λ, et $\forall 1 \leq i \leq n$ une mesure de violation μ_i associée à $\mathtt{c}_i(\mathtt{x})$. La mesure de la violation d'une requête $\mathtt{q}(\mathtt{x})$ est définie par $\mu_{\mathtt{q}}(\mathtt{x}) = \sum_{i=1}^{n} \mu_i(\mathtt{x})$. Le problème de l'extraction de motifs souples pour une requête $\mathtt{q}(\mathtt{x})$ consiste à extraire tous les motifs dont la violation ne dépasse pas le seuil λ, i.e. :

$$Soft(\lambda, \boldsymbol{r}, \mathtt{q}) = \{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mu_{\mathtt{q}}(\mathtt{x}) \leq \lambda\}$$

Le seuil λ ($\lambda \in [0..1]$) permet de contrôler le cumul des violations des contraintes souples de seuil composant la requête $\mathtt{q}(\mathtt{x})$. Ainsi, toute requête $\mathtt{q}(\mathtt{x})$ contenant une ou plusieurs contraintes de seuil $\mathtt{c}_i(\mathtt{x})$ peut être transformée en une requête équivalente ne contenant que des contraintes dures :

— si $\mathtt{c}_i(\mathtt{x})$ est une contrainte *dure*, alors elle reste telle quelle.
— si $\mathtt{c}_i(\mathtt{x})$ est une contrainte *souple*, alors elle est remplacée par sa transformée.

Pour extraire l'ensemble $Soft(\lambda, \boldsymbol{r}, \mathtt{q})$, nous procédons en quatre étapes principales :

1. à chaque contrainte souple de seuil $\mathtt{c}_i(\mathtt{x})$, est associée une sémantique de violation μ_i et une variable de coût z_i permettant de mesurer l'écart au seuil.
2. transformer $\mathtt{c}_i(\mathtt{x})$ en une contrainte dure équivalente $\mathtt{c}'_i(\mathtt{x})$, en utilisant sa relaxation disjonctive (cf. la définition 2.9 à la page 41).
3. ajouter la nouvelle contrainte ($\sum z_i \leq \lambda$) pour contrôler la violation totale (i.e. le cumul des violations).
4. résoudre la requête dure équivalente à l'aide d'un extracteur de motifs fondé sur les CSPs (cf. le chapitre 3).

Les sections suivantes montrent comment toute requête incluant des contraintes souples de seuil peut être transformée en une requête équivalente, uniquement constituée de contraintes dures, pouvant être résolue par un solveur de CSP.

5.1.3a Transformation des contraintes souples de seuil en contraintes dures équivalentes

À l'issue de la première étape, à chaque contrainte souple de seuil $\mathtt{c}_i(\mathtt{x})$ est associée une sémantique de violation μ_i et une variable de coût z_i qui mesure la violation de $\mathtt{c}_i(\mathtt{x})$. En appliquant la relaxation disjonctive (cf. la section 2.5 à la page 39), nous obtenons la transformation, en contraintes dures équivalentes, des contraintes souples de seuil. De plus, compte-tenu de la *forme particulière* de nos contraintes, nous pouvons faire disparaître les disjonctions.

Nous détaillons ci-dessous les différentes transformations obtenues pour chacune des trois sémantiques de violation décrites précédemment ainsi que les reformulations équivalentes déduites, permettant de faire disparaître les disjonctions.

5.1.3a-α Cas de la sémantique de violation μ_1

i) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}})$, la relaxation disjonctive pour μ_1 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}} \wedge z_i = 0] \vee [\mathtt{m}(\mathtt{x}) < \psi_{\mathtt{m}} \wedge z_i = \psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})]$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [z_i = \max(0, \psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x}))]$$

ii) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}})$, la relaxation disjonctive pour μ_1 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}} \wedge z_i = 0] \vee [\mathtt{m}(\mathtt{x}) > \Psi_{\mathtt{m}} \wedge z_i = \mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}}]$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [z_i = \max(0, \mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}})]$$

5.1.3a-β Cas de la sémantique de violation μ_2

i) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}})$, la relaxation disjonctive pour μ_2 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}} \wedge z_i = 0] \vee \left[\mathtt{m}(\mathtt{x}) < \psi_{\mathtt{m}} \wedge z_i = \frac{\psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})}{\psi_{\mathtt{m}}}\right]$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv \left[z_i = \max\left(0, \frac{\psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})}{\psi_{\mathtt{m}}}\right)\right]$$

ii) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}})$, la relaxation disjonctive pour μ_2 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}} \wedge z_i = 0] \vee \left[\mathtt{m}(\mathtt{x}) > \Psi_{\mathtt{m}} \wedge z_i = \frac{\mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}}}{max_{\mathtt{m}} - \Psi_{\mathtt{m}}}\right]$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv \left[z_i = \max\left(0, \frac{\mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}}}{max_{\mathtt{m}} - \Psi_{\mathtt{m}}}\right)\right]$$

5.1.3a-γ Cas de la sémantique de violation μ_3

i) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}})$, la transformation pour μ_3 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \geq (1-\epsilon) \times \psi_{\mathtt{m}}] \wedge \left([\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}} \wedge z_i = 0] \vee \left[\mathtt{m}(\mathtt{x}) < \psi_{\mathtt{m}} \wedge z_i = \frac{\psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})}{\psi_{\mathtt{m}}}\right]\right)$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \geq (1-\epsilon) \times \psi_{\mathtt{m}}] \wedge \left[z_i = \max\left(0, \frac{\psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})}{\psi_{\mathtt{m}}}\right)\right]$$

ii) Soit $\mathtt{c}_i(\mathtt{x}) \equiv (\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}})$, la transformation pour μ_3 est :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \leq (1+\epsilon) \times \Psi_{\mathtt{m}}] \wedge \left([\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}} \wedge z_i = 0] \vee \left[\mathtt{m}(\mathtt{x}) > \Psi_{\mathtt{m}} \wedge z_i = \frac{\mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}}}{\Psi_{\mathtt{m}}}\right]\right)$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'_i(\mathtt{x}) \equiv [\mathtt{m}(\mathtt{x}) \leq (1+\epsilon) \times \Psi_{\mathtt{m}}] \wedge \left[z_i = \max\left(0, \frac{\mathtt{m}(\mathtt{x}) - \Psi_{\mathtt{m}}}{\Psi_{\mathtt{m}}}\right)\right]$$

Ainsi, toute requête contenant une ou plusieurs contraintes de seuil (souples ou dures) $\mathtt{c}_i(\mathtt{x})$ peut être transformée en une requête *équivalente* ne contenant que des contraintes dures.

Enfin, soit λ la quantité maximale de violation autorisée. On définit la variable de coût $z = \sum_{i=1}^{n} z_i$ représentant le cumul des violations et on ajoute la contrainte $z \leq \lambda$.

Ainsi, l'extraction de l'ensemble $Soft(\lambda, \boldsymbol{r}, \mathtt{q})$ (i.e. l'ensemble des motifs souples satisfaisant la requête $\mathtt{q}(\mathtt{x})$), peut être effectuée en résolvant la requête dure correspondante où toutes les contraintes de seuil souples sont transformées en contraintes dures équivalentes et les contraintes dures sont laissées telles quelles. La propriété suivante énonce ce résultat important.

Propriété 2 *Équivalence entre les deux requêtes.* Soient $\mathtt{q}(\mathtt{x}) = \bigwedge_{i=1}^{n} \mathtt{c}_i(\mathtt{x})$ une conjonction de contraintes de seuil $\mathtt{c}_i(\mathtt{x})$, λ la quantité maximale de violation autorisée et z_i la variable de coût associée à $\mathtt{c}_i(\mathtt{x})$ et μ_i sa mesure de violation.
Soit $\mathtt{q}'(\mathtt{x}) = \left[\bigwedge_{i=1}^{n} (z_i = \mu_i(\mathtt{x}))\right] \wedge \left[\sum_{i=1}^{n} z_i \leq \lambda\right]$. Ainsi, nous avons :

$$Soft(\lambda, \boldsymbol{r}, \mathtt{q}) = \mathcal{T}h(\mathcal{L}_{\mathcal{I}}, \boldsymbol{r}, \mathtt{q}')$$

La preuve est immédiate car, chaque contrainte souple $\mathtt{c}_i(\mathtt{x})$ est équivalente à la contrainte dure $(z_i{=}\mu_i(\mathtt{x}))$, et la mesure de violation d'une requête $\mathtt{q}(\mathtt{x})$ est définie comme $\mu_{\mathtt{q}}(\mathtt{x}) = \sum_{i=1}^{n} \mu_i(\mathtt{x})$ (cf. la définition 5.1).

5.1.3b Un cadre flexible pour manipuler la souplesse

Notre approche peut être étendue de plusieurs façons, menant à un cadre plus flexible. Tout d'abord, à chaque contrainte souple de seuil $\mathtt{c}_i(\mathtt{x})$, plusieurs sémantiques de violation peuvent être définies : le gap, la distance relative, etc. En outre, les variables de coûts (z_i) peuvent être exploitées pour un contrôle plus fin de la violation. Par exemple :

- pour limiter la violation d'une contrainte souple de seuil particulière : $z_i \leq \Delta$.
- pour équilibrer la répartition de la violation totale ; celle-ci pouvant être mise en œuvre de deux manières différentes :

  i) pour tout couple de variables de coût, leur différence doit être inférieure à un seuil :

  $$\bigwedge_{1 \leq i < j \leq n} |z_i - z_j| \leq \epsilon$$

  ii) Repartir la quantité de violation pour chacune des contraintes :

  $$\bigwedge_{1 \leq i \leq n} z_i \leq \frac{1}{n} \times \sum_{i=1}^{n} z_i$$

5.1.4 Exemples de mise en œuvre

Dans cette section, nous présentons quelques exemples de mise en œuvre sur deux types de requêtes : locales et n-aires.

5.1.4a Cas d'une requête locale

Considérons à nouveau la requête $\mathtt{q}(\mathtt{x})$ de notre exemple de motivation (cf. la section 5.1.1), et la sémantique de violation μ_2. En appliquant la transformation précédente sur $\mathtt{q}(\mathtt{x})$, nous obtenons la requête dure équivalente suivante :

$$\mathtt{q}'(\mathtt{x}) \equiv \begin{cases} z_1 = \max\left(0, \frac{4-\mathtt{freq(x)}}{4}\right) & \wedge \\ z_2 = \max\left(0, \frac{3-\mathtt{taille(x)}}{3}\right) & \wedge \\ z_3 = \max\left(0, \frac{45-\mathtt{moyenne(x}.prix)}{45}\right) & \wedge \\ z = z_1 + z_2 + z_3 \leq \lambda \end{cases}$$

Ainsi, avec $\lambda = 0$, nous obtenons l'unique solution BDE $\langle 4, 3, 50\rangle$. Avec $\lambda = 0.15$, nous obtenons les quatre motifs suivants. Trois d'entre eux violent le seuil de prix moyen (en gras) : BDE $\langle 4, 3, 50\rangle$, CDE $\langle 4, 3, \mathbf{40}\rangle$, BCE $\langle 4, 3, \mathbf{40}\rangle$ et $BCDE$ $\langle 4, 4, \mathbf{40}\rangle$. Enfin, avec $\lambda = 0.25$, en plus des quatre motifs précédents, nous obtenons un nouveau motif qui viole légèrement la seuil de fréquence (en gras) : BEF $\langle \mathbf{3}, 3, 55\rangle$. Notons que les nouveaux motifs obtenus avec $\lambda = 0.25$ correspondent exactement aux motifs espérés de notre exemple de motivation (cf. la section 5.1.1 à la page 60).

5.1.4b Cas d'une requêtes n-aire

Pour une requête n-aire, nous utilisons le même principe que dans le cas d'une requête locale : pour chaque contrainte n-aire $\mathfrak{C}_i(\mathtt{x})$ de la requête, faire :
— si $\mathfrak{C}_i(\mathtt{x})$ est une contrainte *dure* alors elle reste telle quelle.
— si $\mathfrak{C}_i(\mathtt{x})$ est une contrainte *souple* de seuil alors elle est remplacée par sa transformée.

Pour illustrer le principe de cette transformation sur une requête n-aire, nous avons retenu deux exemples de contraintes n-aires : les règles d'exception et les règles inattendues.

5.1.4b-α Les règles d'exception Comme nous l'avons vu en section 1.4.1-α, les règles d'exception sont composées de 2 parties : une règle générale $(\mathtt{x} \setminus \mathtt{y} \rightarrow \mathtt{j})$ et une règle déviationnelle $(\mathtt{x} \rightarrow \neg\mathtt{j})$. Nous avons choisi de relaxer la règle générale car il est plus facile de fixer les seuils de la règle déviationnelle que les seuils de la règle générale. Nous rappelons ci-dessous la définition de la règle d'exception :

$$\mathtt{exception}(\mathtt{x}, \mathtt{y}, \mathtt{j}) \equiv \begin{cases} \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \geq minfr \wedge \\ \mathtt{freq}(\mathtt{x} \setminus \mathtt{y}) - \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \leq \delta_1 \\ \wedge \\ \mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq maxfr \wedge \\ \mathtt{freq}(\mathtt{x}) - \mathtt{freq}(\mathtt{x} \sqcup \neg\mathtt{j}) \leq \delta_2 \end{cases}$$

$minfr$ et $maxfr$ représentent les seuils de fréquence
δ_1 et δ_2 représentent les seuils de confiance

Ainsi, les contraintes définissant la règle générale sont :

$$\textit{règle générale}(\mathtt{x}, \mathtt{y}, \mathtt{j}) \equiv \begin{cases} \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \geq minfr \wedge \\ \mathtt{freq}(\mathtt{x} \setminus \mathtt{y}) - \mathtt{freq}((\mathtt{x} \setminus \mathtt{y}) \sqcup \mathtt{j}) \leq \delta_1 \end{cases}$$

En appliquant la relaxation disjonctive selon la sémantique de violation μ_3 (cf. la section 5.1.3a-γ), nous obtenons la transformation, en contraintes dures équivalentes, des deux contraintes composant la règle générale :

1. Pour la contrainte $\texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \geq minfr$ (la règle générale est fréquente), la relaxation disjonctive pour μ_3 est :

$$z_1 = \max\left(0, \frac{minfr - \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j})}{minfr}\right) \wedge \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \geq (1-\epsilon) \times minfr$$

2. Pour la contrainte $\texttt{freq}(\texttt{x} \setminus \texttt{y}) - \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \leq \delta_1$ (la règle générale est de forte confiance), la relaxation disjonctive pour μ_3 est :

$$z_2 = \max\left(0, \frac{\texttt{freq}(\texttt{x} \setminus \texttt{y}) - \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) - \delta_1}{\delta_1}\right) \wedge \texttt{freq}(\texttt{x} \setminus \texttt{y}) - \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \leq (1+\epsilon) \times \delta_1$$

Enfin, la requête dure équivalente est donnée ci-dessous :

$$\texttt{exception}(\texttt{x},\texttt{y},\texttt{j}) \equiv \begin{cases} & z_1 = \max\left(0, \frac{minfr - \texttt{freq}((\texttt{x}\setminus\texttt{y})\sqcup\texttt{j})}{minfr}\right) & \wedge \\ & \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \geq (1-\epsilon) \times minfr & \\ \wedge & z_2 = \max\left(0, \frac{\texttt{freq}(\texttt{x}\setminus\texttt{y}) - \texttt{freq}((\texttt{x}\setminus\texttt{y})\sqcup\texttt{j}) - \delta_1}{\delta_1}\right) & \wedge \\ & \texttt{freq}(\texttt{x} \setminus \texttt{y}) - \texttt{freq}((\texttt{x} \setminus \texttt{y}) \sqcup \texttt{j}) \leq (1+\epsilon) \times \delta_1 & \\ \wedge & \texttt{freq}(\texttt{x} \sqcup \neg\texttt{j}) \leq maxfr & \wedge \\ & \texttt{freq}(\texttt{x}) - \texttt{freq}(\texttt{x} \sqcup \neg\texttt{j})) \leq \delta_2 & \\ \wedge & z = z_1 + z_2 \leq \lambda & \end{cases}$$

5.1.4b-β Les règles inattendues Les règles inattendues (cf. la section 1.4.1-β à la page 24) sont définies par :

$$\texttt{inattendue}(\texttt{x},\texttt{y}) \equiv \begin{cases} \texttt{freq}(\texttt{y} \cup \texttt{v}) = 0 \wedge \\ \texttt{freq}(\texttt{x} \cup \texttt{u}) \geq minfr_1 \wedge \\ \texttt{freq}(\texttt{x} \cup \texttt{u} \cup \texttt{y}) \geq minfr_2 \wedge \\ \texttt{conf}(\texttt{x} \cup \texttt{u} \rightarrow \texttt{y}) \geq minconf \wedge \\ (\texttt{freq}(\texttt{x} \cup \texttt{u} \cup \texttt{v}) < maxfr \vee \texttt{conf}(\texttt{x} \cup \texttt{u} \rightarrow \texttt{v}) \leq maxconf) \end{cases}$$

Nous avons choisi de relaxer les trois premières contraintes :

(i) $\texttt{freq}(\texttt{y} \cup \texttt{v}) = 0$;

(ii) $\texttt{freq}(\texttt{x} \cup \texttt{u}) \geq minfr_1$;

(iii) $\texttt{freq}(\texttt{x} \cup \texttt{u} \cup \texttt{y}) \geq minfr_2$;

La relaxation de la contrainte (i) est motivée par le fait qu'il s'agit d'une contrainte dure qui est très difficile à satisfaire dans la pratique. Pour les deux autres contraintes, cela est dû, d'une part, à la difficulté de fixer chacun des deux seuils individuellement pour obtenir les contraintes correspondantes et d'autre part, à la difficulté de donner un seuil approprié à $minfr_1$ par rapport au seuil $minfr_2$.

Nous avons choisi de ne pas relaxer les contraintes $\texttt{conf}(\texttt{x} \cup \texttt{u} \rightarrow \texttt{y}) \geq minconf$ et $(\texttt{freq}(\texttt{x} \cup \texttt{u} \cup \texttt{v}) < maxfr \vee \texttt{conf}(\texttt{x} \cup \texttt{u} \rightarrow \texttt{v}) \leq maxconf)$ car, d'une part, nous souhaitons garder la règle $(\texttt{x} \cup \texttt{u} \rightarrow \texttt{y})$ avec une forte confiance et, d'autre part, la disjonction peut être vue comme une forme de relaxation.

Soit z_1 la variable de coût associée à la violation de la contrainte (i). Sa relaxation disjonctive est définie comme suit :

$$\mathtt{c}'(\mathtt{y} \cup \mathtt{v}) \equiv [\mathtt{freq}(\mathtt{y} \cup \mathtt{v}) = 0 \wedge z_1 = 0] \vee [\mathtt{freq}(\mathtt{y} \cup \mathtt{v}) > 0 \wedge z_1 = \mathtt{freq}(\mathtt{y} \cup \mathtt{v})]$$

Que l'on peut reformuler de manière équivalente par l'unique contrainte :

$$\mathtt{c}'(\mathtt{y} \cup \mathtt{v}) \equiv [z_1 = \mathtt{freq}(\mathtt{y} \cup \mathtt{v})]$$

Pour les deux autres contraintes, nous utilisons la sémantique de violation μ_1 car elles sont de même nature :

— Soit z_2 la variable de coût associée à la violation de la contrainte $\mathtt{freq}(\mathtt{x} \cup u) \geq minfr_1$ ($\mathtt{x} \cup u$ est un motif fréquent). La relaxation disjonctive pour μ_1 est :

$$z_2 = \max(0, minfr_1 - \mathtt{freq}(\mathtt{x} \cup u))$$

— Soit z_3 la variable de coût associée à la violation de la contrainte $\mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{y}) \geq minfr_2$ ($\mathtt{x} \cup \mathtt{u} \rightarrow \mathtt{y}$ est une règle fréquente). La relaxation disjonctive pour μ_1 est :

$$z_3 = \max(0, minfr_2 - \mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{y}))$$

Enfin, la requête dure équivalente est donnée ci-dessous :

$$\mathtt{inattendue}(\mathtt{x}, \mathtt{y}) \equiv \begin{cases} z_1 = \mathtt{freq}(\mathtt{y} \cup \mathtt{v}) \wedge \\ z_2 = \max(0, minfr_1 - \mathtt{freq}(\mathtt{x} \cup \mathtt{u})) \wedge \\ z_3 = \max(0, minfr_2 - \mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{y})) \wedge \\ z = z_1 + z_2 + z_3 \leq \lambda \wedge \\ \mathtt{conf}(\mathtt{x} \cup \mathtt{u} \rightarrow \mathtt{y}) \geq minconf \wedge \\ (\mathtt{freq}(\mathtt{x} \cup \mathtt{u} \cup \mathtt{v}) < maxfr \vee \mathtt{conf}(\mathtt{x} \cup \mathtt{u} \rightarrow \mathtt{v}) < maxconf) \end{cases}$$

5.2 top-k motifs souples

En fouille de données, la recherche des k meilleurs motifs selon une mesure d'intérêt (top-k motifs) se révèle très utile pour trouver les motifs les plus significatifs au regard d'un critère choisi par l'utilisateur. Les méthodes d'extraction des top-k associent à chaque motif un score, puis calculent une liste ordonnée des k meilleurs motifs, i.e. les k motifs ayant les scores les plus élevés [Ke *et al.*, 2009, Wang *et al.*, 2005]. Ces scores sont déterminés par des mesures d'intérêt fournies par l'utilisateur. Jusqu'à présent, les différentes mesures d'intérêt proposées pour l'extraction des top-k motifs ne permettent pas de prendre en compte la souplesse.

Dans cette section, nous définissons la notion de *top-k motifs souples*, i.e. les top-k motifs extraits peuvent inclure des motifs qui violent les contraintes sur les mesures fournies par l'utilisateur. Pour intégrer la souplesse dans le processus d'extraction des top-k, nous proposons une mesure d'intérêt qui exploite notre méthode d'extraction de motifs sous contraintes souples de seuil. L'approche bénéficie pleinement de la manipulation des contraintes souples de seuil.

5.2.1 Exemple de motivation

Reprenons la requête $\mathtt{q}(\mathtt{x}) \equiv \mathtt{freq}(\mathtt{x}) \geq 4 \wedge \mathtt{taille}(\mathtt{x}) \geq 3 \wedge \mathtt{moyenne}(\mathtt{x}.prix) \geq 45$ de notre exemple de motivation (cf. la section 5.1.1). Soit la sémantique de violation μ_2. Considérant les trois motifs

CDE $\langle 4, 3, \mathbf{40}\rangle$, BCE $\langle 4, 3, \mathbf{40}\rangle$ et $BCDE$ $\langle 4, 4, \mathbf{40}\rangle$, obtenus avec $\lambda = 0.15$. Ces trois motifs violent tous le seuil de prix moyen (en gras). Toutefois, avec la sémantique de violation μ_2 (idem pour μ_1 et μ_3), il n'est pas possible de prendre en compte le degré de satisfaction des contraintes de fréquence et de taille. En effet, si un motif $\mathtt{x}'$ satisfait une de ces contraintes alors $\mu_2(\mathtt{x}') = 0$.

Ainsi, dans notre exemple, les trois motifs CDE, BCE et $BCDE$ auront la même importance car ayant le même degré de violation. Or, le motif $BCDE$, dont la taille est légèrement supérieure au seuil $\omega = 3$, peut être considéré comme plus intéressant que les motifs CDE, BCE. D'où l'idée d'introduire une nouvelle mesure permettant de prendre en compte à la fois le degré de satisfaction et d'insatisfaction d'une contrainte souple de seuil.

5.2.2 Intérêt d'un motif pour une contrainte souple de seuil

Une mesure d'intérêt d'un motif pour une contrainte de seuil $\mathtt{c}(\mathtt{x})$ peut être positif (quand $\mathtt{c}(\mathtt{x})$ est satisfaite) ou négative (quand $\mathtt{c}(\mathtt{x})$ n'est pas satisfaite). Comme pour une mesure de violation, une mesure d'intérêt est également normalisée afin de combiner les intérêts de plusieurs contraintes souples de seuil composant la même requête.

Définition 5.2. Intérêt d'un motif pour une contrainte souple de seuil
Étant donné une mesure $\mathtt{m}$, sa valeur maximale $\max_{\mathtt{m}}$ et une mesure d'intérêt $\theta_i : \mathcal{L}_{\mathcal{I}} \rightarrow [-1..1]$. L'intérêt d'un motif $\mathtt{x}$ pour une contrainte $\mathtt{c}_i(\mathtt{x})$, noté $\theta_i(\mathtt{x})$, est définie par :

$$\textit{pour} \quad \mathtt{c}_i(\mathtt{x}) \equiv \mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}} \qquad \theta_i(\mathtt{x}) = \begin{cases} \frac{\mathtt{m}(\mathtt{x}) - \psi_{\mathtt{m}}}{max_{\mathtt{m}} - \psi_{\mathtt{m}}} & si \;\; \mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}} \\ -\mu(\mathtt{x}) & sinon \end{cases}$$

$$\textit{pour} \quad \mathtt{c}_i(\mathtt{x}) \equiv \mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}} \qquad \theta_i(\mathtt{x}) = \begin{cases} \frac{\Psi_{\mathtt{m}} - \mathtt{m}(\mathtt{x})}{\Psi_{\mathtt{m}}} & si \;\; \mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}} \\ -\mu(\mathtt{x}) & sinon \end{cases}$$

5.2.3 Intérêt d'un motif pour un requête

Considérons à présent un ensemble de n mesures m et $\mathtt{q}(\mathtt{x})$ une requête souple, exprimée sous la forme d'une conjonction de n contraintes de seuil (souples ou dures) de la forme $\mathtt{m}(\mathtt{x}) \geq \psi_{\mathtt{m}}$ (ou bien $\mathtt{m}(\mathtt{x}) \leq \Psi_{\mathtt{m}}$). Nous définissons l'intérêt d'un motif $\mathtt{x}$ pour $\mathtt{q}(\mathtt{x})$ comme étant le cumul des intérêts de $\mathtt{x}$ pour les contraintes qui la composent :

$$\theta_{\mathtt{q}}(\mathtt{x}) = \sum_{1 \leq i \leq n} \sigma_i \times \theta_i(\mathtt{x})$$

où σ_i est un coefficient qui traduit l'importance de la contrainte $\mathtt{c}_i(\mathtt{x})$.

Considérons les quatre motifs solutions de la requête $\mathtt{q}(\mathtt{x})$ de la section 5.1.1, obtenus avec la sémantique de violation μ_2 et $\lambda = 0.15$. Supposons à présent que toutes les contraintes composant la requête $\mathtt{q}(\mathtt{x})$ sont de même importance : $\sigma_{\mathtt{freq}} = \sigma_{\mathtt{taille}} = \sigma_{\mathtt{moyenne}} = 1$. Le tableau 5.2 résume les intérêts de ces motifs pour la requête $\mathtt{q}(\mathtt{x})$. Nous pouvons remarquer que les motifs BDE et $BCDE$, qui sont nettement plus intêressants que les autres motifs, ont des scores plus importants.

Supposons maintenant que l'utilisateur veuille donner davantage d'importance à la contrainte de prix moyen. Pour cela, il suffit juste de modifier la valeur du coefficient $\sigma_{\mathtt{moyenne}}$. Par exemple, $\sigma_{\mathtt{freq}} = \sigma_{\mathtt{taille}}$=1

BDE	$\langle 4,3,50\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BDE)$	$=$	$\frac{4-4}{7-4}+\frac{3-3}{6-3}+\frac{50-45}{245-45}$	$=$	0.025
CDE	$\langle 4,3,40\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(CDE)$	$=$	$\frac{4-4}{7-4}+\frac{3-3}{6-3}-\frac{45-40}{45}$	$=$	-0.110
BEF	$\langle 3,3,55\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BEF)$	$=$	$\frac{3-3}{6-3}-\frac{4-3}{4}+\frac{55-45}{245-45}$	$=$	-0.200
$BCDE$	$\langle 4,4,40\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BCDE)$	$=$	$\frac{4-4}{7-4}+\frac{4-3}{6-3}-\frac{45-40}{45}$	$=$	0.223

TABLE 5.2 – Intérêts des motifs pour la requête $\mathtt{q}(\mathtt{x})$, avec $\sigma_{\mathtt{freq}} = \sigma_{\mathtt{taille}} = \sigma_{\mathtt{moyenne}} = 1$.

et $\sigma_{\mathtt{moyenne}} = 2$. Ainsi, comme le montre les résultats du tableau 5.3, les scores des motifs BDE et BEF remontent assez significativement par rapport à ceux du tableau 5.2.

BDE	$\langle 4,3,50\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BDE)$	$=$	0.050
CDE	$\langle 4,3,40\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(CDE)$	$=$	-0.222
BEF	$\langle 3,3,55\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BEF)$	$=$	-0.150
$BCDE$	$\langle 4,4,40\rangle$	$\rightarrow$	$\theta_{\mathtt{q}}(BCDE)$	$=$	0.110

TABLE 5.3 – Intérêts des motifs pour la requête $\mathtt{q}(\mathtt{x})$, avec $\sigma_{\mathtt{freq}} = \sigma_{\mathtt{taille}} = 1$ et $\sigma_{\mathtt{moyenne}} = 2$.

5.2.4 Préférence et top-k motifs souples

Pour extraire les k meilleurs motifs souples, nous définissons une relation de préférence entre les motifs (cf. la définition 1.24 à la page 21), notée $\rhd_{\theta_{\mathtt{q}}}$, permettant d'exploiter la mesure d'intérêt $\theta_{\mathtt{q}}$:

Définition 5.3. RELATION DE PRÉFÉRENCE PAR RAPPORT À $\theta_{\mathtt{q}}$
Étant donné une préférence $\rhd_{\theta_{\mathtt{q}}}$. Un motif $\mathtt{x}$ est strictement préféré à un autre motif $\mathtt{y}$ par rapport à $\theta_{\mathtt{q}}$ (dénoté par $\mathtt{x} \rhd_{\theta_q} \mathtt{y}$) si et seulement si $\theta_{\mathtt{q}}(\mathtt{x}) > \theta_{\mathtt{q}}(\mathtt{y})$ (i.e. $\mathtt{x} \rhd_{\theta_q} \mathtt{y} = \theta_{\mathtt{q}}(\mathtt{x}) > \theta_{\mathtt{q}}(\mathtt{y})$)

En exploitant la définition des top-k motifs (cf. la définition 1.32) et la relation de préférence $\rhd_{\theta_{\mathtt{q}}}$, nous proposons la notion de *top-k motif souple*.

Définition 5.4. TOP-K MOTIF SOUPLE
Étant donné une requête $\mathtt{q}(\mathtt{x})$, la quantité maximale de violation autorisée λ, une relation de préférence $\rhd_{\theta_{\mathtt{q}}}$ [21] et un entier positif k. Un motif $\mathtt{x}$ est un top-k motif souple selon $\rhd_{\theta_{\mathtt{q}}}$ si et seulement s'il n'existe pas plus de $(k-1)$ motifs souples de $Soft(\lambda, \boldsymbol{r}, \mathtt{q})$ dont l'intérêt est plus élevé que celui de $\mathtt{x}$:

$$|\underbrace{\{\mathtt{y} \in Soft(\lambda, \boldsymbol{r}, \mathtt{q}) \mid \ \mathtt{y} \neq \mathtt{x} \wedge \theta_{\mathtt{q}}(\mathtt{y}) > \theta_{\mathtt{q}}(\mathtt{x})\}}_{\lambda\text{-}interesting(\mathtt{x},\theta_{\mathtt{q}},\mathtt{q})}| < k$$

λ-$interesting(\mathtt{x}, \theta_{\mathtt{q}}, \mathtt{q})$ désigne l'ensemble de tous les motifs de $Soft(\lambda, \boldsymbol{r}, \mathtt{q})$ qui sont strictement préférés à $\mathtt{x}$ selon la relation de préférence $\rhd_{\theta_{\mathtt{q}}}$.

Ainsi, le problème d'extraction des top-k motifs pour une requête $\mathtt{q}(\mathtt{x})$ consiste à calculer tous les top-k motifs souples pour $\mathtt{q}(\mathtt{x})$ selon $\rhd_{\theta_{\mathtt{q}}}$. Par exemple, les top-2 motifs souples extraits à partir du tableau 5.2, sont $BCDE$ et BDE.

21. $\rhd_{\theta_{\mathtt{q}}}$ est définie sur l'ensemble $Soft(\lambda, \boldsymbol{r}, \mathtt{q})$.

5.2.5 Calcul des top-k motifs souples

Comme nous allons le voir au chapitre 8, les top-k motifs souples sont un cas particulier de MO (cf. la section 8.1.2i), dont l'extraction sera détaillée en section 8.2.1. L'idée principale est d'exploiter la relation de préférence $\rhd_{\theta_q}$ entre les motifs pour produire un raffinement successif sur les motifs extraits grâce à des contraintes postées dynamiquement au cours de la recherche.

5.3 Travaux relatifs

Il y a très peu de travaux en fouille de données pour faire face à l'aspect dichotomique du cadre habituel de l'extraction de motifs sous contraintes. Dans la littérature, la relaxation a été étudiée pour découvrir des motifs plus inattendus [Antunes et Oliveira, 2004] ou pour offrir des contraintes "relâchées" ayant des propriétés de monotonie dans le but de réutiliser les algorithmes usuels de fouille. Ainsi, dans [Garofalakis *et al.*, 1999], les contraintes d'expressions régulières ont été relâchées en contraintes anti-monotones pour l'extraction des séquences significatives. Dans [Soulet et Crémilleux, 2005], les auteurs proposent de générer automatiquement une *relaxation monotone* et *anti-monotone* d'un ensemble de contraintes.

Dans le cadre de motifs locaux, [Bistarelli et Bonchi, 2007] ont proposé un cadre générique fondé sur les semi-anneau CSP pour exprimer des préférences entre les solutions. Chaque contrainte a sa propre mesure d'intérêt et l'intérêt d'une requête est l'agrégation des intérêts de toutes les contraintes qui composent la requête. Étant donné une requête et une valeur de seuil, le but est de trouver tous les motifs locaux dont l'intérêt ne dépasse cette valeur de seuil. Cependant, cette approche repose sur l'hypothèse forte suivante : *l'intérêt d'une requête satisfait la valeur du seuil, si et seulement si, l'intérêt de chacune des contraintes de la requête satisfait le même valeur de seuil* [Bistarelli et Bonchi, 2007]. Si l'opérateur d'agrégation est effectué en utilisant l'opérateur *min* (*fuzzy semiring*), l'équivalence est préservée. Toutefois, pour l'opérateur $\sum$ (*weighted semiring*) et l'opérateur $\times$ (*probabilistic semiring*), ce n'est plus le cas. C'est pourquoi les auteurs ont besoin d'effectuer une étape de post-traitement pour filtrer l'ensemble des solutions efficaces.

Ainsi, contrairement à [Bistarelli et Bonchi, 2007], notre approche préserve l'équivalence sans nécessiter une étape de post-traitement (cf. la propriété 2). En outre, comme nous l'avons montré, notre approche peut s'appliquer aussi bien sur des motifs locaux que sur des motifs n-aires (cf. la section 5.1.4b).

5.4 Contraintes souples de seuil pour l'extraction de toxicophores

Dans cette section, nous proposons un schéma de relaxation montrant l'intérêt de notre approche sur une application réelle dans le domaine de la chémoinformatique. Ce travail s'inscrit dans une collaboration avec le Centre d'Études et de Recherche sur le Médicament de Normandie (CERMN [22]).

5.4.1 Extraction de toxicophores

La toxicologie est la science étudiant les substances chimiques toxiques. Elle s'intéresse notamment à l'identification de fragments moléculaires [23] spécifiques au sein de la structure d'une molécule appelés

22. `http://www.cermn.unicaen.fr/`

23. Un fragment moléculaire est une sous-structure connexe d'une molécule. Il correspond à un graphe connexe dans le domaine de la fouille de graphes.

toxicophores et considérés comme responsables direct des propriétés toxiques d'une substance chimique. Un objectif majeur est alors d'établir des relations entre de tels fragments et leurs activités afin de mieux identifier les caractéristiques des molécules liées à la toxicité.

Depuis quelques années, plusieurs travaux se sont intéressés à la découverte automatique de fragments moléculaires intéressants. Dans [Auer et Bajorath, 2006, Bajorath, 2008], les auteurs ont introduit la notion de *motif chimique émergent* (ECPs pour Emerging Chemical Patterns). Un ECP correspond à une conjonction de descripteurs moléculaires qui apparaît fréquemment dans une classe de molécules et peu fréquemment dans une autre classe. Leur découverte repose sur un apprentissage automatique effectué à partir d'un ensemble de molécules décrites par des descripteurs moléculaires.

Récemment, G. Poezevara a proposé un nouveau type de motifs, les *motifs émergents de graphes* (EGPs pour Emerging Graph Patterns) [Poezevara *et al.*, 2011]. Ces motifs ont l'intérêt de faire ressortir les contrastes entre deux classes de graphes. La définition de ces motifs provient directement de celle des motifs émergents dans le cas des données ensemblistes (cf. le taux de croissance). L'extraction de ces motifs est fondée sur l'enchaînement d'une technique de recherche de sous-graphes fréquents utilisée pour changer la description des données avec une méthode récente de fouille sous contraintes dans le cas de données binaires. Les contraintes de fréquence et d'émergence (cette dernière étant issue de la mesure de taux de croissance) définissent ces motifs qui s'avèrent précieux pour la prédiction de la toxicité [Poezevara *et al.*, 2010].

Afin d'extraire des motifs émergents qui peuvent être considérés comme de possibles toxicophores, nous proposons de combiner les contraintes d'émergence et de fréquence, caractérisant les motifs du point de vue de leur présence dans les données, avec des connaissances chimiques, comme l'`aromaticité` ou la `rigidité` d'une molécule, qui sont des indicateurs connus de la toxicité (cf. la section 5.4.2). Dans ce travail, nous nous intéressons à la découverte de fragments toxicophores sur des molécules décrites par des attributs correspondant à des sous-graphes fréquents initialement extraits des molécules et pour lesquels il est possible d'attacher des valeurs de propriétés chimiques, comme l'`aromaticité` ou la `rigidité`.

5.4.2 Contraintes de seuil considérées

Une difficulté majeure de la tâche est le nombre potentiel de motifs qui est très grand. Il devient alors important de réduire le nombre de motifs extraits à ceux présentant un intérêt potentiel exprimé par l'utilisateur sous forme de contraintes. Ci-dessous une description des différentes contraintes que nous avons retenues :

- **L'émergence** permet de caractériser une molécule d'une classe (toxique) par rapport à une autre classe (non-toxique). Les motifs émergents traduisent l'hypothèse toxicophore (**H**1) : si une molécule possède dans sa structure les fragments moléculaires d'un motif émergent, alors elle possède des caractéristiques de toxicité et est donc particulièrement susceptible d'être toxique. L'émergence est mesurée par le taux de croissance (cf. la définition 1.13 à la page 15). Soit ρ un seuil minimal pour le taux de croissance. On impose la contrainte souple de seuil :

$$\mathtt{c}_1(\mathtt{x}) \equiv \mathtt{émergence}_\mathtt{j}(\mathtt{x}) \geq \rho$$

- **Contrainte de fréquence** : les motifs avec une fréquence très faible sont souvent dûs à des artefacts dans les données et constituent du bruit. Soit $minfr$ un seuil minimal pour la fréquence. Afin d'assurer une représentativité de l'information extraite, on impose la contrainte souple de seuil :

$$\mathtt{c}_2(\mathtt{x}) \equiv \mathtt{freq}(\mathtt{x}) \geq minfr$$

- **Contrainte d'aromacité** : pour chaque sous-graphe (donc chaque attribut) est associé une valeur de l'aromaticité qui est une mesure chimique. L'intérêt de cette mesure est qu'elle véhicule une

hypothèse toxicophore (**H2**) : plus un attribut a une forte valeur d'aromaticité, plus une molécule supportant cet attribut tend à être toxique. L'aromaticité d'un motif est la moyenne de l'aromaticité de ses attributs. Soit ψ_{arom} un seuil minimal pour l'aromaticité. Pour extraire des motifs intégrant une connaissance chimique portant sur l'aromaticité de ces attributs, on impose la contrainte souple de seuil :

$$\mathtt{c}_3(\mathtt{x}) \equiv \mathtt{aromaticité}(\mathtt{x}) \geq \psi_{arom}$$

- **Contrainte de rigidité** : plus un sous-graphe est rigide, plus ce sous-graphe est "solide" ; un motif composé d'attributs rigides renforce l'hypothèse d'un toxicophore (**H3**). La rigidité d'un motif est la moyenne des rigidités de ses sous-graphes [24]. Soit $\psi_{\mathtt{rigidité}}$ le seuil minimal de rigidité. On impose la contrainte souple de seuil :

$$\mathtt{c}_4(\mathtt{x}) \equiv \mathtt{rigidité}(\mathtt{x}) \geq \psi_{\mathtt{rigidité}}$$

La requête soumise $\mathtt{q}(\mathtt{x})$ est la conjonction des quatre contraintes souples de seuil présentées ci-dessus :

$$\mathtt{q}(\mathtt{x}) \equiv \mathtt{c}_1(\mathtt{x}) \wedge \mathtt{c}_2(\mathtt{x}) \wedge \mathtt{c}_3(\mathtt{x}) \wedge \mathtt{c}_4(\mathtt{x})$$

Comme nous pouvons le constater, nous avons plusieurs mesures dont il est difficile de donner les seuils pour obtenir les contraintes correspondantes et il est encore plus difficile de donner un seuil approprié pour une mesure par rapport aux autres seuils donnés. D'où l'intérêt de relâcher les seuils, ce qui donne, de façon relative, moins d'importance au choix des seuils. Enfin, en ce qui concerne les sémantiques de violation, on peut a priori faire plus confiance à la connaissance du domaine (notamment l'aromaticité) et donc pénaliser plus fortement une violation de la contrainte liée à l'aromaticité que les autres contraintes.

5.4.3 Transformation en une requête dure équivalente

Pour cette requête, nous avons retenu la sémantique de violation μ_2 (i.e. l'écart relatif), car les différentes contraintes composant la requête sont de nature hétérogène (avis donné par les chimistes impliqués dans le travail), d'où l'idée de normaliser les écarts. En appliquant la relaxation disjonctive selon la sémantique de violation μ_2 (cf. la section 5.1.3a-γ), nous obtenons la requête dure équivalente suivante :

$$q'(\mathtt{x}) \equiv \begin{cases} z_1 = \max\left(0, \frac{\rho - \mathtt{émergence_j(x)}}{\rho}\right) & \wedge \\ z_2 = \max\left(0, \frac{minfr - \mathtt{freq(x)}}{minfr}\right) & \wedge \\ z_3 = \max\left(0, \frac{\psi_{arom} - \mathtt{aromaticité(x)}}{\psi_{arom}}\right) & \wedge \\ z_4 = \max\left(0, \frac{\psi_{\mathtt{rigidité}} - \mathtt{rigidité(x)}}{\psi_{\mathtt{rigidité}}}\right) & \wedge \\ z = z_1 + z_2 + z_3 + z_4 \leq \lambda \end{cases}$$

λ représente la quantité maximale de violation autorisée et z le cumul des violations, où z_i est la variable de coût associée à chaque contrainte souple de seuil $\mathtt{c}_i(\mathtt{x})$.

5.4.4 CSP issu de la transformation

Dans cette section, nous montrons comment la relaxation disjonctive de la requête $\mathtt{q}(\mathtt{x})$ peut se modéliser sous forme d'un CSP.

24. La rigidité d'un sous-graphe est égale à $2e/v(v-1)$, où e (resp. v) est le nombre de ses arêtes (resp. sommets).

Le CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ modélisant la requête $\mathtt{q}'(\mathtt{x})$ est le suivant :

a) $\mathcal{X} = \{\mathtt{x}\} \bigcup\limits_{1 \leq i \leq 4} \{z_i\} \cup \{z\}$ est l'ensemble des variables, avec :

— $\mathtt{x}$, la variable ensembliste représentant le motif recherché,

— $\{z_1, z_2, z_3, z_4\}$, les variables de coût quantifiant les violations des contraintes souples de seuil,
— z, la variable de coût quantifiant la violation globale.

b) $\mathcal{D} = D_{\mathtt{x}} \bigcup\limits_{1 \leq i \leq 4} \{D_{z_i}\} \cup \{D_z\}$, l'ensemble des domaines, avec :
— $D_{z_i} = [0..1]$
— $D_z = [0..\lambda]$ $(\lambda \in [0..1])$

c) $\mathcal{C} = \mathcal{C}'_{num} \cup \{z = \sum z_i\} \cup \{z \leq \lambda\}$, l'ensemble des contraintes, avec :

$$\mathcal{C}'_{num} = \{ \quad z_1 = \max\left(0, \frac{\rho - \texttt{émergence}_1(\mathtt{x})}{\rho}\right), z_2 = \max\left(0, \frac{minfr - \texttt{freq}(\mathtt{x})}{minfr}\right), \\ z_3 = \max\left(0, \frac{\psi_{arom} - \texttt{aromaticité}(\mathtt{x})}{\psi_{arom}}\right), z_4 = \max\left(0, \frac{\psi_{\texttt{rigidité}} - \texttt{rigidité}(\mathtt{x})}{\psi_{\texttt{rigidité}}}\right)\}$$

5.5 Expérimentations sur les jeux de données de l'*UCI*

Dans cette section, nous montrons la faisabilité et les apports pratiques de l'introduction de la souplesse pour l'extraction des règles d'exception (cf. la section 5.5.2) et des règles inattendues (cf. la section 5.5.3), sur les jeux de données de l'*UCI*.

5.5.1 Protocole expérimental

Différentes expérimentations ont été menées sur plusieurs jeux de données de l'*UCI* ainsi que sur un jeu de données réel (noté Meningitis) issu de de l'Hôpital Central de Grenoble. Meningitis recense les pathologies des enfants atteints d'une méningite virale ou bactérienne. Le tableau 5.4 résume les différentes caractéristiques des jeux de données utilisés.

Toutes nos expérimentations ont été réalisées sur un processeur Intel core i3 à 2.13 GHz ayant 4 Go de RAM. La mise en œuvre de notre approche a été réalisée en *Gecode* (cf. l'annexe A.1).

Jeu de données	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{I}\rvert$	densité
Mushroom	8124	119	0.18
Soybean	630	50	0.32
Tic-tac-toe	958	29	0.33
Meningitis	329	84	0.25

TABLE 5.4 – Description des jeux de données de l'*UCI* retenus pour nos expérimentations.

Comme la résolution effectuée par le solveur de contraintes est correcte et complète, notre approche permet d'extraire l'ensemble correct et complet des motifs satisfaisant les requêtes dures équivalentes associées à la relaxation des règles d'exception et inattendues (cf. la section 5.1.4b).

5.5.2 Règles d'exception

Pour introduire la relaxation dans l'extraction de règles d'exception, nous avons retenu la sémantique de violation μ_3 (i.e. écart relatif restreint). Comme cette sémantique de violation nécessite l'introduction d'un paramètre ϵ, appelé *écart de violation*, nous avons distingué deux cas :

— même écart de violation pour toutes les contraintes souples de seuil (un seul paramètre ϵ donné par l'utilisateur pour toutes les contraintes) ;

— différents écarts de violation, un pour chaque contrainte souple de seuil (un paramètre ϵ_i donné par l'utilisateur pour chaque contrainte).

Les valeurs des différents seuils ont été choisis de sorte à rendre le problème d'extraction sur-contraint pour mieux mesurer l'impact de la relaxation.

5.5.2a Même écart de violation pour toutes les contraintes souples de seuil

La figure 5.1 montre l'évolution du nombre de paires de règles d'exception extraites en fonction des paramètres λ (seuil de violation maximale aurorisée) et ϵ (écart de violation) pour les jeux de données soybean et meningitis. Nous avons aussi testé d'autres jeux de données et comme les résultats obtenus sont similaires, ils ne sont pas reportés.

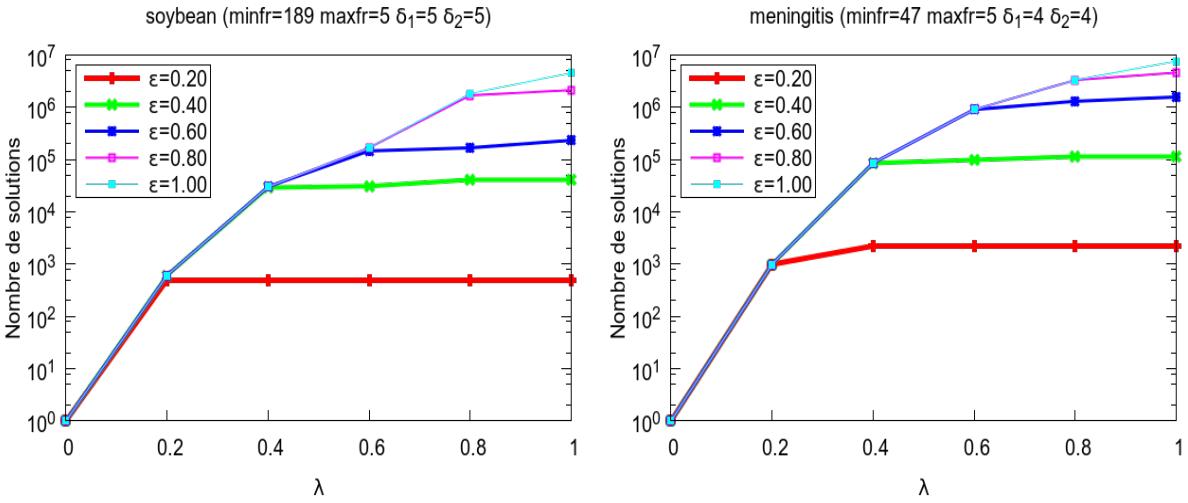


FIGURE 5.1 – Évolution du nombre de règles d'exception extraites avec le même écart de violation.

Pour nos expérimentations, nous avons initialisé λ à ϵ (i.e. $\lambda = \epsilon$) pour 2 raisons principales :

— minimiser le nombre de paramètres fixés par l'utilisateur. En effet, en plus des seuils $minfr$, $maxfr$, δ_1 et δ_2, l'utilisateur doit également définir la valeur du paramètre ϵ.

— imposer à chaque contrainte d'être violée au plus par un écart ϵ et que la violation globale ne peut dépasser ce même écart.

Notons que pour $\epsilon = 0$, nous nous ramenons au cas dur (i.e. aucune contrainte n'est relaxée). Dans ce cas, comme nous pouvons le constater sur la figure 5.1, il n'existe pas solutions satisfaisant les contraintes (avec des seuils durs) de la règle d'exception. En revanche, en relaxant les seuils de fréquence et de confiance de la règle générale, nous arrivons à trouver des solutions "proches".

Comme attendu, plus la valeur de ϵ est grande, plus le nombre de règles d'exception augmente. En effet, quand ϵ augmente, le seuil de fréquence (resp. de confiance) de la règle générale diminue (resp.

augmente) et il y a donc un plus grand nombre de règles générales. Notons enfin que la courbe devient une ligne droite pour ($\lambda > \epsilon$). Cela est dû au fait que la violation maximale autorisée est bornée par ϵ.

5.5.2b Différents écarts de violation pour les contraintes souples de seuil

Pour ce second cas, nous avons réalisé deux séries d'expérimentations. Tout d'abord, nous autorisons uniquement la relaxation du seuil de fréquence de la règle générale (i.e. $\epsilon_2 = 0$). Puis, nous relaxons uniquement le seuil de confiance (i.e. $\epsilon_1 = 0$).

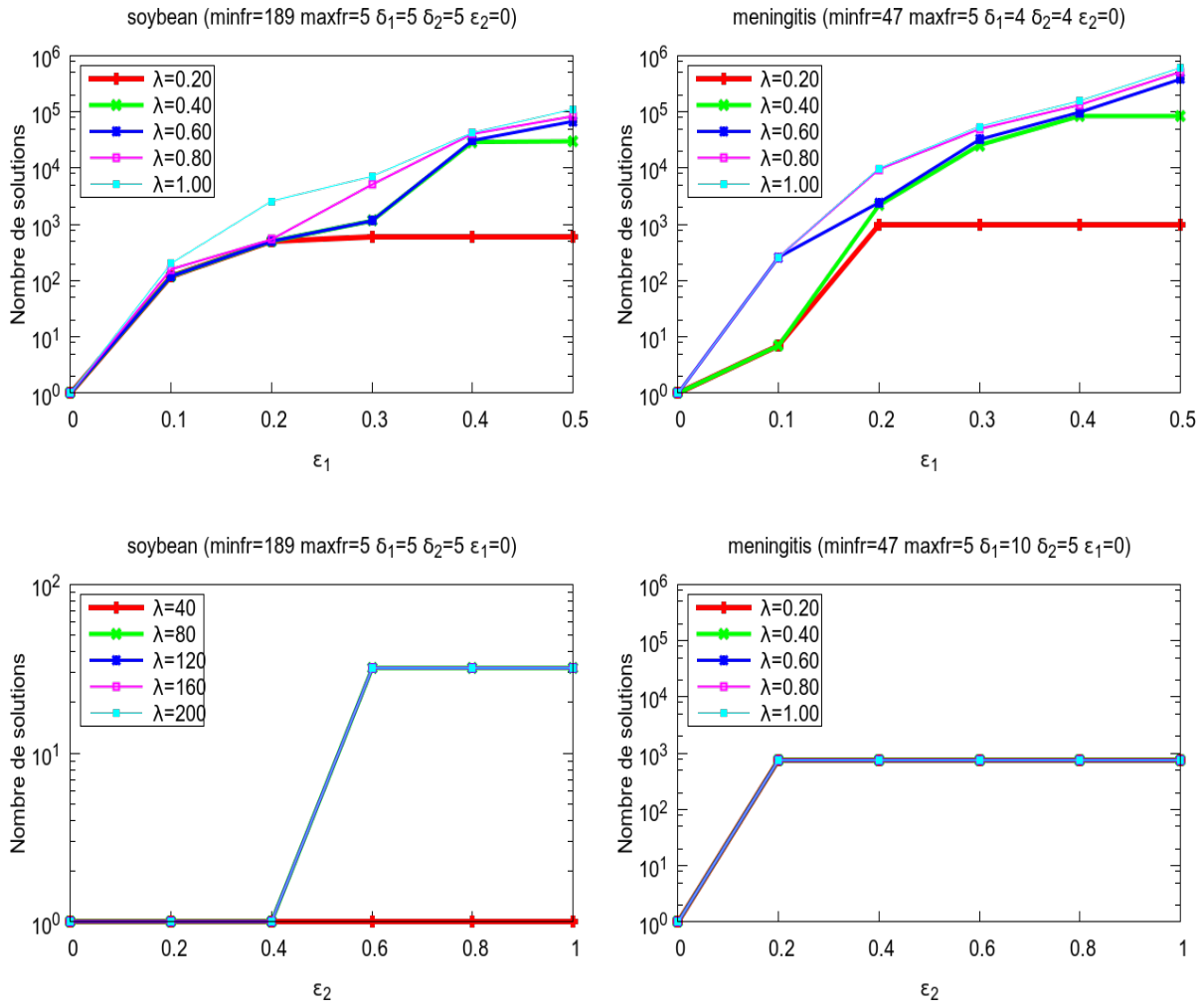


FIGURE 5.2 – Évolution du nombre de règles d'exception extraites avec différents écarts de violation.

La figure 5.2 montre l'évolution du nombre de paires de règles d'exception extraites en fonction des valeurs des paramètres λ et ϵ_i ($i = \{1, 2\}$), pour les jeux de données soybean et meningitis. Une fois de plus, nous retrouvons le même comportement que précédemment, à savoir, plus ϵ_i est grand, plus le nombre de règles d'exception augmente. Par ailleurs, nous constatons que plus la valeur de λ est grand, plus la courbe représentant cette valeur est dominante.

Sur le jeu de données soybean, nous pouvons remarquer que l'écart de violation associé au seuil de confiance (paramètre ϵ_2) est beaucoup plus important dans le cas d'un seuil de fréquence dur (i.e. $\epsilon_1 = 0$). Ce qui n'est pas le cas pour un seuil de confiance dur (i.e. $\epsilon_2 = 0$), où de petits écarts de violation du paramètre ϵ_1 sont suffisants pour trouver des premières solutions.

5.5.2c Discussions

Les règles d'exception sont un cas particulier des règles rares [Khiari *et al.*, 2010b]. Même s'il existe quelques travaux permettant d'extraire les règles rares [Szathmary *et al.*, 2007], il est impossible de distinguer les règles d'exception à partir de l'ensemble des règles rares. C'est une limitation forte car la plupart des règles rares sont peu fiables, d'où l'intérêt des règles d'exception et de leur extraction. Savoir rechercher directement les règles d'exception permet de réduire de manière drastique le nombre de motifs obtenus.

Toutefois, pour certaines requêtes, il est difficile d'obtenir des solutions, car l'utilisateur ne connaît pas *a priori* les bonnes valeurs pour les seuils, d'où l'intérêt de trouver des solutions "*proches*" pour pouvoir tirer de conclusions de ces solutions, à la différence du cadre dur pour lequel on n'aurait aucune solution.

5.5.3 Règles inattendues

Pour introduire la relaxation dans l'extraction des règles inattendues, nous avons choisi d'utiliser la sémantique de violation μ_1 (écart absolu), car les trois contraintes relaxées sont toutes des contraintes de fréquence, donc il n'est pas nécessaire de normaliser l'écart pour pouvoir agréger les valeurs des variables de coût z_i.

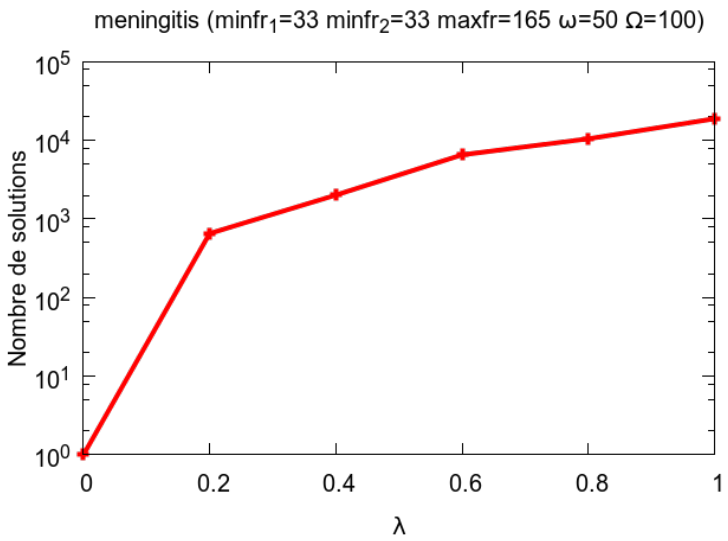


FIGURE 5.3 – Évolution du nombre de règles inattendues extraites.

Les expérimentations ont été menées sur le jeu de données réel meningitis (cf. le tableau 5.4). La figure 5.3 décrit l'évolution du nombre de paires de règles inattendues en fonction du seuil λ (seuil de violation maximale). Pour $\lambda = 0$ (aucune contrainte n'est relaxée), aucune solution n'a pu être trouvée sur ce jeu de données. Avec notre approche de relaxation, il est alors possible de trouver des solutions plus

proches des valeurs des seuils. Comme attendu, plus λ est grand, plus le nombre de règles inattendues augmente. En effet, quand λ augmente, les fréquences des contraintes relaxées diminuent, par conséquent le nombre de règles inattendues augmente.

5.5.3a Discussions

Les règles inattendues sont un cas particulier des règles rares. À la différence des règles d'exception, où l'on cherche une règle qui contredit une règle générale (qui est cherché au même temps), dans les règles inattendues, on cherche une règle qui contredit une croyance $\mathtt{u} \rightarrow v$, i.e. une règle donnée par l'utilisateur (avec un paramètre γ_{belief}).

D'après les conclusions dressées par M. Khiari lors de ces travaux portant sur l'extraction des règles inattendues en utilisant une modélisation CSP [Khiari *et al.*, 2010b], il est difficile de trouver des solutions, même pour des jeux de données petits. D'où l'intérêt d'utiliser la relaxation de contraintes pour trouver des solutions "*proches*".

5.6 Expérimentations sur les jeux de données ECB

Dans cette section, nous montrons, au travers de l'application à la découverte de fragments moléculaires toxicophores (cf. la section 5.4), comment les contraintes souples de seuil permettent d'extraire des motifs (très) pertinents qui n'auraient pas pu être découverts avec des seuils durs.

5.6.1 Protocole expérimental

Nous avons utilisé un jeu de données (base European Chemicals Bureau) préparé par le CERMN. La toxicité est fondée sur l'indicateur quantitatif de toxicité $CL50$ [25]. En fonction de la valeur de cet indicateur, les molécules sont réparties dans les trois catégories suivantes : $H400$ très toxique ($CL50 \leq 1$ mg/L), $H401$ toxique (1 mg/L $< CL50 \leq 10$ mg/L), et $H402$ nocif (10 mg/L $< CL50 \leq 100$ mg/L).

Dans cette étude, nous nous concentrons uniquement sur les classes $H400$ et $H402$. Le jeu de données $\boldsymbol{r}$ utilisé contient 567 molécules, 372 de la classe $H400$ et 195 de la classe $H402$. Les molécules sont représentées en utilisant $1,450$ sous-graphes connexes fréquents initialement extraits de $\boldsymbol{r}$ [26] au seuil de fréquence de 1%.

Pour l'aromaticité et la rigidité, les seuils ont été fixés à environ 2/3 de leur valeur maximale ($\psi_{\texttt{aromaticité}}$=60 et $\psi_{\texttt{rigidité}} = 60$). En effet, des seuils élevés sont en faveur des hypothèses toxicophores (**H**2 et **H**3 cf. la section 5.4.2). Pour l'émergence et la fréquence, les seuils ont été fixés à environ 1/4 de leur valeur maximale ($\rho = 5$ et $minfr = 90$) de manière à obtenir un compromis entre la fréquence et le taux de croissance. Un tel choix nous permet d'extraire uniquement les motifs les plus fréquents ayant les meilleurs taux de croissance. Nous avons considéré trois valeurs différentes pour la quantité maximale de violation autorisée : $\lambda(\%) \in \{0\%, 20\%, 40\%\}$. Pour évaluer l'intérêt des motifs extraits, nous avons fixé $\sigma_{\texttt{émergence}_i}$, $\sigma_{\texttt{freq}}$ et $\sigma_{\texttt{rigidité}}$ à 1 et $\sigma_{\texttt{aromaticité}}$ à 2. En effet, l'aromaticité constitue une connaissance chimique plus importante.

Les motifs extraits sont constitués de fragments moléculaires et pour évaluer la présence de toxicophores dans leur description, une analyse d'experts a conduit à l'identification de toxicophores (environ-

25. Concentration létale d'une substance dans l'eau nécessaire pour causer la mort de 50% d'une population dans des conditions expérimentales précises.

26. Une molécule Ch contient un item A si Ch supporte A, et A est un sous-graphe connexe fréquent de $\boldsymbol{r}$.

nementaux) bien connus, à savoir le *benzène*, le *phénol*, les chloro-substitués aromatiques (par exemple *chlorobenzène*), les *organophosphorés*, les amines aromatiques (par exemple *l'aniline*), le *pyrrole*, et les hydrocarbures aromatiques polycycliques (par exemple *naphtalène*).

5.6.2 Extraction de motifs émergents souples

Le tableau 5.5 présente les résultats obtenus pour la requête `q(x)` (cf. la section 5.4.2). Il indique les nombres de motifs émergents (souples) extraits contenant au moins un toxicophore dans son intégralité (colonnes notées **T**) ou des sous-fragments d'un toxicophore (colonnes notées **F**) parmi cinq des six fragments moléculaires répertoriés dans la base, ceci pour les trois valeurs de λ retenues (cf. la section 5.6.1).

	Total						top-25						top-50					
λ (%)	0		20		40		0		20		40		0		20		40	
# Solutions	**1,066**		**7,109**		**278,360**		**25**						**50**					
Temps de calcul	**14m42s**		**15m33s**		**19m14s**													
Toxicophore (SMILES)	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
Benzene c1ccccc1	330	1,053	3,357	6,845	101,379	269,847	2	23	**4**	21	**6**	18	8	42	**16**	34	**13**	36
Phenol c1(ccccc1)O	287	878	2,900	6,151	37,347	224,709	4	9	**6**	5	**2**	3	5	18	**10**	11	**9**	7
Chlorobenzene Clc1ccccc1	0	143	346	703	486	930	0	10	**6**	6	**2**	2	0	24	**9**	15	**6**	8
Pyrrole c1cncc1						1						1						1

TABLE 5.5 – Répartition des motifs émergents souples en fonction des toxicophores connus pour `q(x)`.

Cette répartition est donnée pour le nombre total de solutions trouvées (colonnes 2-7) mais aussi pour les *top*-25 (colonnes 8-13) et les *top*-50 (colonnes 14-19). Comme les deux catégories **T** et **F** ne sont pas disjointes, la somme de leurs nombres de motifs est toujours supérieure ou égale au nombre total de solutions (# solutions). Le temps nécessaire pour extraire l'ensemble de toutes les solutions est d'environ 14 minutes pour ($\lambda = 0\%$), 15 minutes pour ($\lambda = 20\%$) et 19 minutes pour ($\lambda = 40\%$).

Comme le montre les résultats du tableau 5.5, 47%[27](resp. 36.4%) des motifs émergents souples extraits avec $\lambda = 20\%$ (resp. 40%) contiennent du benzène (fragment de type **T**), contre environ 31% pour $\lambda = 0\%$. Les seuils souples permettent ainsi de mieux retrouver ce toxicophore (gain moyen d'environ 11%). Pour les fragments de type **F**, la proportion de solutions extraites contenant des sous-fragments du benzène (code smiles[28]) : {`cc`, `ccc`, `cccc`, `ccccc`}) est presque identique dans les cas dur et le cas souple (environ 97%). Cette tendance se confirme sur le phénol, où 40% des solutions extraites avec $\lambda = 20\%$ contiennent un tel fragment, contre 26.9% pour $\lambda = 0\%$. Pour $\lambda = 40\%$, le ratio de motifs émergents souples extraits contenant du phénol est d'environ 13.4%. Une fois de plus, les seuils souples permettent de mieux détecter ce toxicophore, en particulier avec $\lambda = 20\%$ (gain d'environ 13%).

Pour le chlorobenzène (avec $\lambda = 0\%$), seuls les motifs contenant des fragments moléculaires de type **F** sont extraits : {`Clc(c)cc`, `Clc(c)ccc`, `Clc(c)cccc`, `Clc(cc)ccc`, `Clccc` ...}. Les seuils souples permettent de trouver en moyenne 2.5% de toxicophores contenant le chlorobenzène (i.e. fragment de type **T**). En outre, pour les composés aromatiques contenant de l'azote, de nouveaux motifs avec une nouvelle caractéristique chimique (contenant le sous-fragment `nc`) sont découverts. En effet, ce dérivé, qui n'est pas détecté avec ($\lambda = 0$), est assez difficile à extraire car il est associé à un fragment chimique ayant une faible valeur de fréquence.

27. Ratio entre le nombre de solutions contenant un toxicophore par le nombre total de solutions.
28. Code smiles est une notation de ligne pour décrire la structure des molécules chimiques : `http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html`

Les motifs émergents souples contenant le cycle aromatique de l'aniline ne sont pas détectés en raison de leur faible rigidité (33). En effet, avec $\lambda = 40\%$, la valeur minimale autorisée est de $60 \times 0.60 = 36$. Une augmentation très légère de λ ($\lambda = 45\%$), permettrait l'extraction de ces motifs émergents.

Enfin, les motifs organo-phosphorés sont caractérisés par une très forte émergence ($+\infty$). Ce sont des JEPs (cf. la définition 1.14). Les chimistes ont un fort intérêt pour de tels motifs. Ils ne figurent pas dans le tableau 5.5 et sont traités à part dans la section 5.6.4.

5.6.3 Extraction des top-k motifs émergents souples

Les résultats du tableau 5.5 montrent que parmi les *top*-25 (resp. *top*-50) motifs émergents durs extraits avec $\lambda = 0\%$, uniquement 2 (resp. 4) motifs contiennent le benzène (resp. phénol). Les autres *top-k* motifs émergents sont constitués uniquement de sous-fragments de chlorobenzène.

Le tableau 5.6 donne les *top*-25 motifs émergents souples extraits avec $\lambda = 20\%$. Les lignes en jaune correspondent à des motifs obtenus avec $\lambda = 0\%$ et ayant au moins un cycle complet de phénol, tandis que les lignes en gris correspondent aux nouveaux motifs extraits grâce aux contraintes souples de seuils (les contraintes violées sont sont mises en évidence en noir).

Les seuils souples permettent de retrouver 4 nouveaux motifs émergents souples contenant du phénol parmis les *top*-25 motifs souples (lignes 10-13), ce qui représente un ratio de 1.5 ($\lambda = 20\%$ détecte 1.5 fois plus de motifs émergents utiles comparé à $\lambda = 0\%$). Notons que deux de ces motifs contiennent également un cycle aromatique (le benzène) (lignes 10 et 12). En outre, ces motifs, qui violent légèrement la contrainte de rigidité, sont très aromatiques et d'un point de vue de la biodégradabilité des composés

N	Intérêt	Motif	Émergence	Fréquence	Aromaticité	Rigidité	SMILES
1	193	425 483 763	7	101	95	66	cc ccc c1(ccccc1)O
2	191	160 425 483	8	89	95	66	Clc1ccccc1 cc ccc
3	189	425 483 566 763	7	101	96	62	cc ccc cccc c1(ccccc1)O
4	187	160 425 483 566	8	89	96	62	Clc1ccccc1 cc ccc cccc
5	185	425 483 493	5	118	90	72	cc ccc cccO
6	184	119 425 483	9	94	92	68	Clcccc cc ccc
7	184	119 425 483 566	9	94	94	64	Clcccc cc ccc cccc
8	183	425 483 500	6	104	92	68	cc ccc ccc(c)O
9	183	425 483 500 566	6	104	94	64	cc ccc ccc(c)O cccc
10	183	425 483 763 821	7	101	96	59	cc ccc c1(ccccc1)O c1ccccc1
11	183	425 483 732 763	7	101	96	59	cc ccc ccccc c1(ccccc1)O
12	183	425 483 566 763 821	7	101	97	57	cc ccc cccc c1(ccccc1)O c1ccccc1
13	183	425 483 566 732 763	7	101	97	57	cc ccc cccc ccccc c1(ccccc1)O
14	183	120 425 483	9	93	93	66	Clc(c)ccc cc ccc
15	183	120 425 483 566	9	93	95	62	Clc(c)ccc cc ccc cccc
16	183	159 425 483	9	92	93	66	Clccccc cc ccc
17	183	159 425 483 566	9	92	95	62	Clccccc cc ccc cccc
18	182	425 483 736	6	103	93	66	cc ccc ccc(c)cc
19	182	425 483 566 736	6	103	95	62	cc ccc cccc ccc(c)cc
20	182	425 483 626	6	103	93	66	cc ccc cccc(c)O
21	182	425 483 566 626	6	103	95	62	cc ccc cccc cccc(c)O
22	181	160 425 483 821	8	89	96	59	Clc1ccccc1 cc ccc c1ccccc1
23	181	160 425 483 732	8	89	96	59	Clc1ccccc1 cc ccc ccccc
24	181	160 425 483 566 821	8	89	97	57	Clc1ccccc1 cc ccc cccc c1ccccc1
25	181	160 425 483 566 732	8	89	97	57	Clc1ccccc1 cc ccc cccc ccccc

TABLE 5.6 – *top*-25 motifs émergents souples extraits avec $\lambda = 20\%$.

aromatiques sont parmi les plus récalcitrants des polluants. Ces motifs ont un taux de croissance élevé et ce résultat renforce notre hypothèse (**H**1) que la mesure du taux de croissance reflète le comportement toxique. De plus, $\lambda = 20\%$ conduit à l'extraction de 6 nouvelles motifs émergents souples contenant du chlorobenzène. Notons que deux de ces motifs (lignes 2 et 4) violent un peu la contrainte de fréquence, tandis que les quatre autres (lignes 22-25) violent à la fois les contraintes de fréquence et de rigidité. Ces motifs sont d'un grand intérêt et ils renforcent nos hypothèses précédentes de toxicophore.

Le tableau 5.7 représente les *top*-25 motifs émergents souples extraits avec $\lambda = 40\%$. Comme précédemment, les seuils souples permettent de retrouver 6 de nouveaux motifs émergents souples contenant du benzène (cf. les lignes 7, 9, 12, 14, 15 et 16). Ces motifs, qui violent légèrement la contrainte de taux de croissance, sont fortement aromatiques et relativement rigides, ce qui et renforce l'hypothèse que plus la rigidité chimique est élevée, plus dangereux est son comportement dans l'environnement (**H**3). Un nouvel motif émergent d'un intérêt particulier pour les chimistes est obtenu : $\{nc\}$. Ce motif est dangereuse pour l'environnement, car il correspond au composé aromatique azoté qui sont souvent toxiques pour les espèces aquatiques.

Pour les *top*-50 motifs émergents souples, les seuils souples $\lambda = 20\%$ (resp. 40%) permettent de détecter 2 (resp. 1.8) fois plus de motifs contenant du phénol. De plus, $\lambda = 40\%$ permet d'extraire 13 (resp. 6) nouveaux motifs émergents souples contenant du benzène (resp. le chlorobenzène).

Tous ces résultats confirment le bénéfice d'utiliser des seuils souples afin d'obtenir de nouvelles connaissances chimiques d'un grand intérêt.

N	Intérêt	Motif	Émergence	Fréquence	Aromaticité	Rigidité	SMILES
1	301	425	3	289	100	100	cc
2	275	203	7	65	100	100	nc
3	258	425 483	3	288	100	83	cc ccc
4	237	425 566	3	281	100	75	cc cccc
5	230	425 483 566	3	281	100	72	cc ccc cccc
6	224	425 732	3	279	100	70	cc ccccc
7	223	425 821	3	274	100	70	cc c1ccccc1
8	219	425 483 732	3	279	100	68	cc ccc ccccc
9	218	425 483 821	3	274	100	68	cc ccc c1ccccc1
10	216	483	3	288	100	66	ccc
11	209	425 483 566 732	3	279	100	64	cc ccc cccc ccccc
12	208	425 483 566 821	3	274	100	64	cc ccc cccc c1ccccc1
13	206	425 566 732	3	279	100	63	cc cccc ccccc
14	205	425 566 821	3	274	100	63	cc cccc c1ccccc1
15	200	425 483 732 821	3	274	100	61	cc ccc ccccc c1ccccc1
16	198	425 732 821	3	274	100	60	cc ccccc c1ccccc1
17	193	425 483 763	7	101	95	66	cc ccc c1(ccccc1)O
18	191	160 425 483	8	89	95	66	Clc1ccccc1 cc ccc
19	189	425 483 566 763	7	101	96	62	cc ccc cccc c1(ccccc1)O
20	187	160 425 483 566	8	89	96	62	Clc1ccccc1 cc ccc cccc
21	185	425 483 493	5	118	90	72	cc ccc cccO
22	184	119 425 483	9	94	92	68	Clcccc cc ccc
23	184	119 425 483 566	9	94	94	64	Clcccc cc ccc cccc
24	183	425 483 500	6	104	92	68	cc ccc ccc(c)O
25	183	425 483 500 566	6	104	94	64	cc ccc ccc(c)O cccc

TABLE 5.7 – *top*-25 motifs émergents souples extraits avec $\lambda = 40\%$.

5.6.4 Extraction des top-k Jumping Emerging Patterns souples

Notre troisième série d'expérimentations évalue le caractère de toxicité porté par les fragments chimiques présents uniquement dans les molécules classées H400 (très toxiques), autrement dit les Jumping Emerging Patterns (JEPs) (cf. la définition 1.14). Le tableau 5.8 présente les *top*-25 JEPs extraits pour différentes valeurs de λ. On peut dresser les remarques suivantes : montre les *top*-25

(i) Sans les seuils souples, aucun JEP n'a pu être extrait.

(ii) Avec $\lambda = 50\%$ (resp. 60% et 70%), nous obtenons 3 (resp. 218 et 421, 504) JEPs souples. En effet, ces motifs sont peu fréquents, il est donc nécessaire d'avoir un seuil de violation relativement élevé.

(iii) Tous les motifs contenant des sous-fragments organophosphorés ont un taux de croissance infini. Ce composant est une généralisation de plusieurs Jumping Emerging Fragments et peut être vu comme une une sorte de *structure commune maximale* de ces fragments.

(iv) Parmi les *top*-25 JEPs souples extraits avec $\lambda = 60\%$, les motifs les plus intéressants sont ceux contenant un *cycle benzénique* ({`c1ccccc1`}). Avec $\lambda = 50\%$, les JEPs souples extraits contiennent des sous-fragments de benzène (i.e. sans cycle complet). Ces JEPs sont moins pertinents d'un point de vue chimique par rapport à ceux extraits avec $\lambda = 60\%$,

(v) L'augmentation de la valeur de λ à 70% conduit à la détection de plusieurs nouveaux JEPs souples prometteurs. Ces JEPs, qui contiennent la fonction amine (par exemple l'aniline {`c1(ccccc1)N`}), sont très toxiques pour les organismes aquatiques.

A nouveau, ces résultats montrent la pertinence et l'apport des contraintes souples de seuil et des top-k motifs souples pour mettre en évidence les structures chimiques les plus parlantes, comme par exemple les tels que les cycles benzéniques par rapport à leurs sous-fragments.

5.7 Conclusion

Dans ce chapitre, nous avons proposé un cadre générique et flexible pour relaxer les contraintes de seuil. Nous avons également introduit la notion du top-k motifs souples. Notre approche peut s'appliquer aussi bien sur des motifs locaux que sur des motifs impliquant plusieurs motifs locaux (i.e. les motifs n-aires, les top-k motifs, etc). Enfin, la pertinence et l'efficacité de notre approche a été mise en évidence par des expérimentations sur des jeux de données de l'*UCI* et une étude de cas dans le domaine de la chemoinformatique pour l'extracation de toxicophores.

N	Intérêt	Motif	Émergence	Fréquence	Aromaticité	Rigidité	SMILES
$\lambda = 50\%$							# **Solutions=3**
1	153	425 483 896	∞	47	66	88	cc ccc OP
2	122	425 483 1050	∞	45	66	77	cc ccc COP
3	122	425 483 914	∞	45	66	77	cc ccc OPO

$\lambda = 60\%$											# **Solutions=218**				
1	174	425	483	566	732	896	∞	42	80	71	cc	ccc	cccc	ccccc	OP
2	174	425	483	566	732	821	∞	40	83	66	cc	ccc	cccc	ccccc	c1ccccc1
		896									OP				
3	172	425	483	566	821	896	∞	40	80	71	cc	ccc	cccc	c1ccccc1	OP
4	168	425	483	566	896		∞	42	75	79	cc	ccc	cccc	OP	
5	167	425	483	732	821	896	∞	40	80	69	cc	ccc	ccccc	c1ccccc1	OP
6	161	425	483	732	896		∞	42	75	76	cc	ccc	ccccc	OP	
7	160	425	566	732	821	896	∞	40	80	66	cc	cccc	ccccc	c1ccccc1	OP
8	159	425	483	821	896		∞	40	75	76	cc	ccc	c1ccccc1	OP	
9	157	425	483	566	732	821	∞	38	83	60	cc	ccc	cccc	ccccc	c1ccccc1
		1050									COP				
10	157	425	483	566	732	821	∞	38	83	60	cc	ccc	cccc	ccccc	c1ccccc1
		914									OPO				
11	155	425	483	566	732	1050	∞	40	80	64	cc	ccc	cccc	ccccc	COP
12	155	425	483	566	732	914	∞	40	80	64	cc	ccc	cccc	ccccc	OPO
13	153	425	483	896			∞	47	66	88	cc	ccc	OP		
14	153	425	483	566	821	1050	∞	38	80	64	cc	ccc	cccc	c1ccccc1	COP
15	153	425	483	566	821	914	∞	38	80	64	cc	ccc	cccc	c1ccccc1	OPO
16	151	425	566	732	896		∞	42	75	72	cc	cccc	ccccc	OP	
17	149	425	566	821	896		∞	40	75	72	cc	cccc	c1ccccc1	OP	
18	148	425	483	732	821	1050	∞	38	80	62	cc	ccc	ccccc	c1ccccc1	COP
19	148	425	483	732	821	914	∞	38	80	62	cc	ccc	ccccc	c1ccccc1	OPO
20	145	425	483	566	732	916	∞	38	80	61	cc	ccc	cccc	ccccc	OPOC
21	144	483	566	732	821	896	∞	40	80	59	ccc	cccc	ccccc	c1ccccc1	OP
22	144	425	732	821	896		∞	40	75	70	cc	ccccc	c1ccccc1	OP	
23	144	425	483	566	1050		∞	40	75	70	cc	ccc	cccc	COP	
24	144	425	483	566	914		∞	40	75	70	cc	ccc	cccc	OPO	
25	142	425	566	732	821	1050	∞	38	80	59	cc	cccc	ccccc	c1ccccc1	COP

$\lambda = 70\%$											# **Solutions=421,504**				
1	192	172	425	483	566		∞	27	92	61	Clcc(ccc)Cl	cc	ccc	cccc	
2	184	172	425	483			∞	27	89	64	Clcc(ccc)Cl	cc	ccc		
3	181	119	425	483	740		∞	28	90	59	Clcccc	cc	ccc	c1(ccccc1)N	
4	180	120	425	483	735		∞	29	90	58	Clc(c)ccc	cc	ccc	ccc(cc)N	
5	176	119	425	483	735		∞	29	89	59	Clcccc	cc	ccc	ccc(cc)N	
6	175	160	377	425	483	566	∞	29	88	61	Clc1ccccc1	cN	cc	ccc	cccc
		732	821								ccccc	c1ccccc1			
7	174	105	425	483	566	732	∞	28	88	61	Clc	cc	ccc	cccc	ccccc
		740	821								c1(ccccc1)N	c1ccccc1			
8	174	425	483	566	732	896	∞	42	80	71	cc	ccc	cccc	ccccc	OP
9	174	425	483	566	732	821	∞	40	83	66	cc	ccc	cccc	ccccc	c1ccccc1
		896									OP				
10	173	119	377	425	483	566	∞	30	85	66	Clcccc	cN	cc	ccc	cccc
		732									ccccc				
11	173	160	377	425	483	566	∞	29	86	64	Clc1ccccc1	cN	cc	ccc	cccc
		821									c1ccccc1				
12	173	160	377	425	483	566	∞	29	86	64	Clc1ccccc1	cN	cc	ccc	cccc
		732									ccccc				
13	172	105	425	483	566	740	∞	28	86	64	Clc	cc	ccc	cccc	c1(ccccc1)N
		821									c1ccccc1				
14	172	105	425	483	566	732	∞	28	86	64	Clc	cc	ccc	cccc	ccccc
		740									c1(ccccc1)N				
15	172	119	377	425	483	566	∞	29	85	66	Clcccc	cN	cc	ccc	cccc
		821									c1ccccc1				
16	172	119	377	425	483	566	∞	29	87	62	Clcccc	cN	cc	ccc	cccc
		732	821								ccccc	c1ccccc1			
17	172	425	483	566	821	896	∞	40	80	71	cc	ccc	cccc	c1ccccc1	OP
18	171	119	377	425	483	566	∞	30	82	71	Clcccc	cN	cc	ccc	cccc
19	170	160	377	425	483	732	∞	29	86	63	Clc1ccccc1	cN	cc	ccc	ccccc
		821									c1ccccc1				
20	169	105	425	483	732	740	∞	28	86	63	Clc	cc	ccc	ccccc	c1(ccccc1)N
		821									c1ccccc1				
21	169	120	377	425	483	566	∞	29	87	61	Clc(c)ccc	cN	cc	ccc	cccc
		732	821								ccccc	c1ccccc1			
22	169	159	377	425	483	566	∞	29	87	61	Clccccc	cN	cc	ccc	cccc
		732	821								ccccc	c1ccccc1			
23	169	160	377	425	483	566	∞	29	83	69	Clc1ccccc1	cN	cc	ccc	cccc
24	168	105	425	483	566	732	∞	28	87	61	Clc	cc	ccc	cccc	ccccc
		735	821								ccc(cc)N	c1ccccc1			
25	168	105	425	483	566	740	∞	28	83	69	Clc	cc	ccc	cccc	c1(ccccc1)N

TABLE 5.8 – *top*-25 Jumping Emerging Patterns souples extraits avec $\lambda = 50\%$, 60%, et 70%.

"Je veux être ce que j'ai toujours voulu être : dominant."

Tiger Woods

Chapitre 6

Skypatterns souples

Sommaire

La notion de skyline, présentée au chapitre 4, a été récemment intégrée dans l'extraction de motifs sous contraintes pour définir la notion de skypattern [Soulet *et al.*, 2011]. Par exemple, un utilisateur peut préférer les motifs de fréquence élevée, de grande taille et de grande confiance. Dans ce cas, un motif `x` domine un autre motif `y` si et seulement si $\texttt{freq}(\texttt{y}) \geq \texttt{freq}(\texttt{x})$, $\texttt{taille}(\texttt{y}) \geq \texttt{taille}(\texttt{x})$ et $\texttt{conf}(\texttt{y}) \geq \texttt{conf}(\texttt{x})$, où au moins une inégalité stricte est vérifiée. Étant donné un ensemble de motifs, l'ensemble des skypatterns contient les motifs qui ne sont dominés par aucun autre motif.

Les skypatterns sont intéressants à double titre : ils ne nécessitent pas de seuil pour les contraintes portant sur les mesures ; de plus, la notion de dominance offre un intérêt global avec une sémantique facilement compréhensible par l'utilisateur. Néanmoins, les skypatterns, comme les autres motifs satisfaisant une requête donnée, souffrent de l'aspect dichotomique du cadre fondé sur les contraintes. En effet, un motif satisfait ou ne satisfait pas une propriété ou une contrainte. Mais, les motifs qui sont proches seront ignorés.

La contribution de ce chapitre est double. Tout d'abord, nous introduisons la notion de skypattern souple. Puis, nous montrons comment le problème de l'extraction de skypatterns (durs ou souples) peut être modélisée et résolue en utilisant les CSP dynamiques.

Ce chapitre est organisé comme suit. La section 6.1 présente la notion de skypattern. La section 6.2 introduit la notion de skypattern souple. La section 6.3 montre comment extraire, aussi bien les skypatterns que les skypatterns souples, à l'aide des CSP dynamiques. Les sections 6.4 et 6.5 sont consacrées aux expérimentations : la première fait du benchmarking sur les jeux de données de l'*UCI*, alors que la seconde décrit notre étude de cas consacrée aux toxicophores.

6.1 Skypatterns

Les skypatterns ont été récemment introduits par [Soulet *et al.*, 2011]. Ces motifs permettent à l'utilisateur d'exprimer des préférences à l'aide d'une relation de Pareto dominance : étant donné un ensemble de mesures M, les skypatterns sont les motifs pour lesquels aucune mesure de M ne peut être améliorée sans dégrader les autres. Par exemple, un utilisateur peut préférer les motifs de fréquence élevée, de grande taille et de grande confiance. Dans ce cas, un motif $\mathtt{x}$ domine un autre motif $\mathtt{y}$ pour l'ensemble de mesures $M = \{\mathtt{freq}, \mathtt{taille}, \mathtt{conf}\}$ si et seulement si $\mathtt{freq}(\mathtt{y}) \geq \mathtt{freq}(\mathtt{x})$, $\mathtt{taille}(\mathtt{y}) \geq \mathtt{taille}(\mathtt{x})$ et $\mathtt{conf}(\mathtt{y}) \geq \mathtt{conf}(\mathtt{x})$, où au moins une inégalité stricte est vérifiée.

6.1.1 Définitions

Soit un ensemble de motifs P, et un ensemble de mesures M. Si un motif $\mathtt{x} \in P$ est dominé par un autre motif $\mathtt{y} \in P$ selon M, alors le motif $\mathtt{x}$ ne sera pas retenu, car jugé moins intéressant que le motif $\mathtt{y}$. L'ensemble des skypatterns de P contient les motifs (de P) qui ne sont dominés par aucun autre motif (selon M).

De façon similaire à la définition 4.1, on définit :

Définition 6.1. Pareto dominance
Étant donné un ensemble de mesures M, un motif $\mathtt{x}$ domine un autre motif $\mathtt{y}$ par rapport à M (dénoté par $\mathtt{x} \succ_M \mathtt{y}$) si et seulement si $\forall m \in M, m(\mathtt{x}) \geq m(\mathtt{y})$ et $\exists m' \in M, m'(\mathtt{x}) > m'(\mathtt{y})$.

Considérons l'exemple décrit par le tableau 6.1. En partie gauche, chaque transaction t_i est associée à un client i, et chaque item (dénoté A,...,F) d'une transaction correspond à un produit acheté par ce client. En partie droite, un attribut (*prix*) est associé à chaque produit.

Trans.	Items					
t_1		B			E	F
t_2		B	C	D		
t_3	A				E	F
t_4	A	B	C	D	E	
t_5		B	C	D	E	
t_6		B	C	D	E	F
t_7	A	B	C	D	E	F

Item	Prix
A	30
B	40
C	10
D	40
E	70
F	55

Table 6.1 – Jeu de données transactionnel $\boldsymbol{r}$.

Soit l'ensemble de mesures $M = \{\mathtt{freq}, \mathtt{aire}\}$. Le motif BCD domine le motif BC car $\mathtt{freq}(BCD) = \mathtt{freq}(BC) = 5$ et $\mathtt{aire}(BCD) > \mathtt{aire}(BC)$. Pour $M = \{\mathtt{freq}, \mathtt{taille}, \mathtt{moyenne}\}$, le motif BDE do-

mine le motif BCE car $\texttt{freq}(BDE) = \texttt{freq}(BCE) = 4$, $\texttt{taille}(BDE) = \texttt{taille}(BCE) = 3$ et $\texttt{moyenne}(BDE.prix) > \texttt{moyenne}(BCE.prix)$.

Définition 6.2. Opérateur skypattern
Étant donné un ensemble de motifs P et un ensemble de mesures M, un skypattern de P, par rapport à M, est un motif de P non dominé selon M. L'opérateur $Sky(M, P)$ détermine tous les skypatterns de P selon M :

$$Sky(M, P) = \{\mathtt{x} \in P \mid \nexists\, \mathtt{y} \in P, \mathtt{y} \succ_M \mathtt{x}\}$$

Reprenons l'exemple décrit par le tableau 6.1. Soit $M = \{\texttt{freq}, \texttt{taille}\}$, alors $Sky(M, \mathcal{L}_\mathcal{I}) = \{ABCDEF,\ BCDEF, ABCDE, BCDE, BCD, B, E\}$. La figure 6.1 représente graphiquement ces skypatterns, avec comme abscisse la mesure `freq` et comme ordonnée la mesure `taille`.

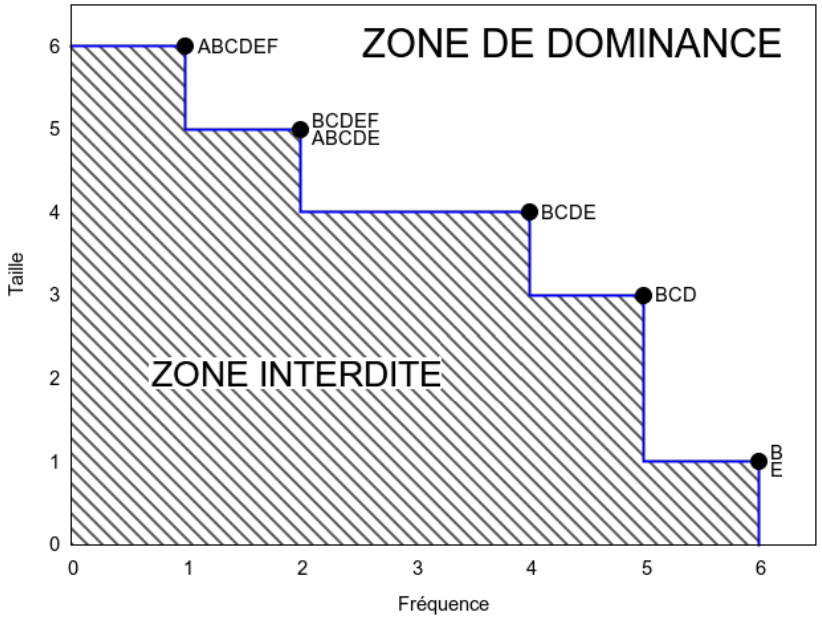


Figure 6.1 – Skypatterns extraits de l'exemple du tableau 6.1.

La zone hachurée est appelée *zone interdite* (cf. la figure 6.1) car elle ne peut contenir aucun skypattern. L'autre partie constitue la *zone de dominance*. La ligne bleue (frontière Pareto) représente la limite entre ces deux zones.

Les skypatterns $BCDEF$ et $ABCDE$ sont représentés par un même point sur la figure 6.1. En effet, ces deux motifs ont la même fréquence et sont de même taille. Il en est de même pour les motifs B et E. Les motifs ayant les mêmes valeurs pour un ensemble de mesures M seront appelés motifs indistincts par rapport à M (cf. la section 7.1 à la page 112). Ils joueront un rôle important dans la construction du cube de skypatterns présentée au chapitre 7 (cf. page 111).

6.1.2 Extraction des skypatterns : `Aetheris`

Extraire les skypatterns est un problème bien différent d'extraire les skylines (cf. la section 4.2 à la page 53). En effet, tout skyline d'un ensemble de points $\mathcal{E}$ appartient à $\mathcal{E}$. Par contre, un skypattern

appartient à $\mathcal{L}_\mathcal{I}$ où $\mathcal{I}$ est un ensemble d'items et $\mathcal{T}$ un ensemble de transactions sur $\mathcal{I}$. Extraire les skypatterns est clairement plus difficile qu'extraire les skylines car l'espace de recherche pour les skypatterns $\mathcal{O}(2^{|\mathcal{I}|})$ est beaucoup plus grand que l'espace de recherche pour les skylines $\mathcal{O}(|\mathcal{E}|)$.

[Soulet *et al.*, 2011] ont proposé AETHERIS une approche dont l'efficacité est obtenue en tirant profit des relations théoriques entre les représentations condensées de motifs et les skypatterns. Les auteurs montrent que tout skypattern selon M peut être obtenu à partir d'une représentation condensée des motifs selon M. Une idée majeure d'AETHERIS est d'identifier automatiquement un petit ensemble de mesures $M' \subseteq M$ (où M' est appelé "ensemble de mesures skylineables de M") et de montrer que tout skypattern selon M peut être obtenu à partir de la représentation condensée selon M'. La production des skypatterns s'obtient avec une étape où l'opérateur Sky selon M est appliqué sur les motifs de la représentation condensée. Comme la représentation condensée selon M' comporte (beaucoup) moins de motifs que celle selon M, il s'ensuit un gain de temps pour l'extraction des motifs.

Enfin, à notre connaissance, AETHERIS est le seul et unique extracteur de skypatterns développé par la communauté fouille de données.

6.2 Skypatterns souples

Cette section définit la notion de skypattern souple qui constitue la première contribution de ce chapitre. Comme nous l'avons mentionné en introduction, les skypatterns souffrent eux aussi de l'aspect dichotomique du cadre fondé sur les contraintes. Puisque les skypatterns sont les sommets de la frontière Pareto, l'idée est de capturer les motifs situés sur cette frontière ou très proches de celle-ci. A cet effet, nous proposons deux types de skypatterns souples :

1. les *edge-skypatterns* qui appartiennent à la frontière Pareto (sans en être forcément un sommet) (cf. la section 6.2.1)
2. les *δ-skypatterns* qui sont très proches (au sens du paramètre δ) de la frontière Pareto (cf. la section 6.2.2).

Pour assouplir la notion de skypatterns, il nous a fallu renforcer la notion de dominance. En effet, plus la dominance sera restrictive, plus il y aura de motifs non-dominés. C'est pourquoi nous avons introduit la notion de dominance stricte et de δ-dominance.

6.2.1 Edge-skypatterns

Les skypatterns ont été définis comme les motifs non dominés au sens d'une Pareto dominance. De manière analogue, les edge-skypatterns sont définis à partir de la notion de dominance stricte.

Définition 6.3. Dominance stricte
Étant donné un ensemble de mesures M, un motif $\mathtt{x}$ domine strictement un autre motif $\mathtt{y}$ par rapport à M (dénoté par $\mathtt{x} \gg_M \mathtt{y}$) si et seulement si $\forall m \in M$, $m(\mathtt{x}) > m(\mathtt{y})$.

Définition 6.4. Opérateur Edge-skypattern
Étant donné un ensemble de motifs P et un ensemble de mesures M, un edge-skypattern de P, par rapport à M, est un motif de P non strictement dominé par rapport à M. L'opérateur $Edge\text{-}Sky(M, P)$ détermine tous les edge-skypatterns de P par rapport à M :

$$Edge\text{-}Sky(M, P) = \{\mathtt{x} \in P \mid \nexists\, \mathtt{y} \in P, \mathtt{y} \gg_M \mathtt{x}\}$$

La figure 6.2 représente les $28 = 7 + (4 + 8 + 3 + 4 + 2)$ edge-skypatterns extraits de l'exemple du tableau 6.1 pour $M = \{\texttt{freq}, \texttt{taille}\}$. Tous les edge-skypatterns appartiennent à la frontière Pareto, et sept d'entre eux sont des skypatterns, i.e. situés sur les sommets de cette frontière (cf. la figure 6.1).

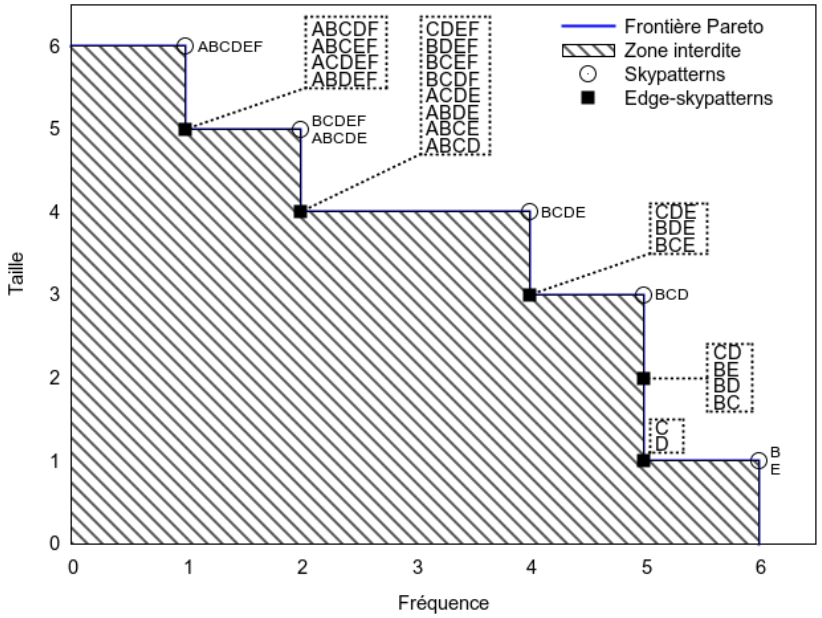


FIGURE 6.2 – Edge-skypatterns extraits de l'exemple du tableau 6.1.

Le motif BE est un bien un edge-skypattern pour $M = \{\texttt{freq}, \texttt{taille}\}$ car il est non-strictement dominé. En effet, la fréquence de BE est de 5 et sa taille de 2. Les seuls motifs pouvant dominer strictement BE sont B et E, qui sont tous deux de fréquence 6. Mais, ces deux motifs n'ont pas une taille strictement plus grande que 2. Enfin, le motif BE n'est pas un skypattern car il est dominé (au sens de $\succ_M$) par le motif BCD ayant la même fréquence mais une taille strictement supérieure.

Propriété 3 . Soient deux motifs $\texttt{x}$ et $\texttt{y}$, $\texttt{x} \gg_M \texttt{y} \implies \texttt{x} \succ_M \texttt{y}$.

Démonstration. Soit $\texttt{x}$ et $\texttt{y}$ deux motifs tels que $\texttt{x} \gg_M \texttt{y}$

$$\begin{array}{ll} \forall m \in M : m(\texttt{x}) > m(\texttt{y}) & \text{(Définition 6.3)} \\ \forall m \in M : m(\texttt{x}) \geq m(\texttt{y}) \bigwedge \exists m' \in M : m'(\texttt{x}) > m'(\texttt{y}) & \\ \text{Alors, } \texttt{x} \succ_M \texttt{y} & \text{(Définition 6.1)} \end{array} \quad (6.1)$$

□

Théorème 1 . Soit M un ensemble de mesures, et P un ensemble de motifs.

$$Sky(M, P) \subseteq Edge\text{-}Sky(M, P)$$

Démonstration. Soit $\mathtt{x} \in Sky(M, P)$

$$\begin{array}{ll} \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \succ_M \mathtt{x} & \text{(Définition 6.2)} \\ \forall \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \not\succ_M \mathtt{x} & \\ \forall \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \not\gg_M \mathtt{x} & \text{(Contraposée de l'équation (6.1))} \\ \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \gg_M \mathtt{x} & \\ \mathtt{x} \in \textit{Edge-Sky}(M, P) & \text{(Définition 6.4)} \end{array} \tag{6.2}$$

□

6.2.2 δ-skypatterns

Les δ-skypatterns sont définis de manière analogue aux edge-skypatterns en introduisant la notion de δ-dominance, pour $0 < \delta \leq 1$. Les δ-skypatterns sont des motifs proches de la frontière Pareto, la valeur de δ exprimant la distance relative maximale entre un skypattern et cette frontière.

Définition 6.5. δ-**Dominance**
Étant donné un ensemble de mesures M, un motif $\mathtt{x}$ δ-domine un autre motif $\mathtt{y}$ par rapport à M (dénoté $\mathtt{x} \succ^{\delta}_{M} \mathtt{y}$), si et seulement si $\forall m \in M, (1-\delta) \times m(\mathtt{x}) > m(\mathtt{y})$.

Définition 6.6. **Opérateur** δ-**skypattern**
Étant donné un ensemble de motifs P et un ensemble de mesures M, un δ-skypattern de P par rapport à M est un motif de P non δ-dominé par rapport à M. L'opérateur $\delta\text{-}Sky(M, P)$ détermine tous les δ-skypatterns par rapport à M :

$$\delta\text{-}Sky(M, P) = \{\mathtt{x} \in P \mid \nexists\, \mathtt{y} \in P, \mathtt{y} \succ^{\delta}_{M} \mathtt{x}\}$$

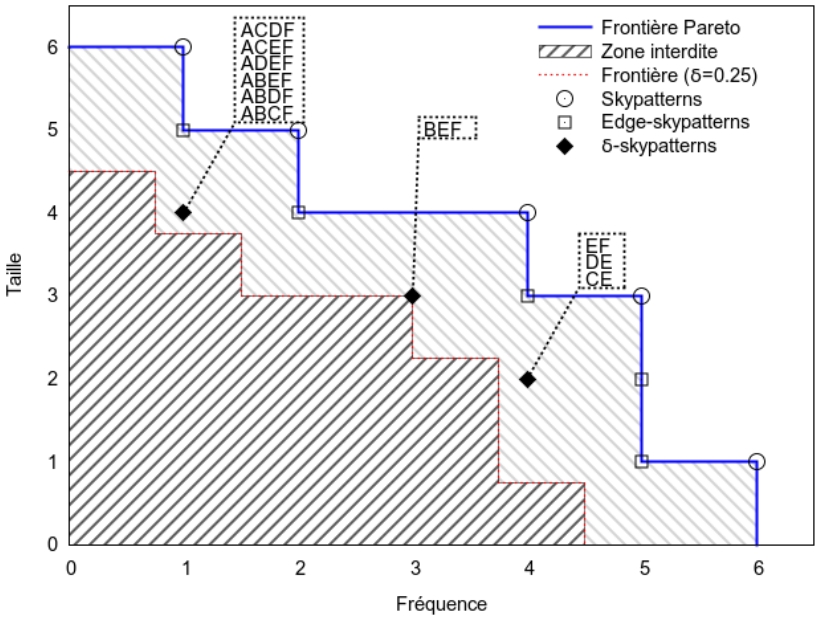


Figure 6.3 – δ-skypatterns (qui ne sont pas des edge-skypatterns) extraits de l'exemple du tableau 6.1.

Il y a 38 = (28+10) δ-skypatterns extraits de l'exemple dans le tableau 6.1 pour $M = \{\texttt{freq}, \texttt{taille}\}$ et $\delta = 0.25$. La figure 6.3 ne décrit que les 10=6+3+1 δ-skypatterns qui ne sont pas des edge-skypatterns. Intuitivement, les δ-skypatterns sont des motifs proches de la frontière Pareto, la valeur de δ exprimant la distance relative maximale entre un skypattern et cette frontière.

Propriété 4 . Soient $\mathtt{x}$ et $\mathtt{y}$ deux motifs, $\mathtt{x} \succ^{\delta}_{M} \mathtt{y} \implies \mathtt{x} \gg_{M} \mathtt{y}$.

Démonstration. Soient $\mathtt{x}$ et $\mathtt{y}$ deux motifs tels que $\mathtt{x} \succ^{\delta}_{M} \mathtt{y}$

$$\begin{array}{ll} \forall m \in M : (1-\delta) \times m(\mathtt{x}) \geq m(\mathtt{y}) & \text{(Définition 6.5)} \\ \forall m \in M : m(\mathtt{x}) > m(\mathtt{y}) & \text{(comme } \delta > 0\text{)} \\ \text{Alors, } \mathtt{x} \gg_{M} \mathtt{y} & \text{(Définition 6.3)} \end{array} \tag{6.3}$$

□

Théorème 2 . Soit M un ensemble de mesures, et P un ensemble de motifs.

$$\textit{Edge-Sky}(M, P) \subseteq \delta\textit{-Sky}(M, P)$$

Démonstration. Soit $\mathtt{x} \in \textit{Edge-Sky}(M, P)$

$$\begin{array}{ll} \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \gg_{M} \mathtt{x} & \text{(Définition 6.4)} \\ \forall \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \not\gg_{M} \mathtt{x} & \\ \forall \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \not\succ^{\delta}_{M} \mathtt{x} & \text{(Contraposée de l'équation (6.3))} \\ \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \succ^{\delta}_{M} \mathtt{x} & \\ \mathtt{x} \in \delta\textit{-Sky}(M, P) & \text{(Définition 6.6)} \end{array} \tag{6.4}$$

□

Pour conclure, soit M un ensemble de mesures, et P un ensemble de motifs. On a la double inclusion suivante :

$$\textit{Sky}(M, P) \subseteq \textit{Edge-Sky}(M, P) \subseteq \delta\textit{-Sky}(M, P)$$

Enfin, pour $\delta = 0$, les δ-skypatterns correspondent aux edge-skypatterns :

$$\textit{Edge-Sky}(M, P) = \delta\textit{-Sky}(M, P)$$

6.3 Extraction à l'aide des CSP dynamiques

Cette section présente la seconde contribution de ce chapitre. Elle décrit comment l'extraction des skypatterns durs et l'extraction des skypattern souples peuvent être modélisées et résolues, de manière analogue, à l'aide des CSP dynamiques (cf. la section 2.6 à la page 42).

L'idée majeure de notre approche consiste à agrandir successivement la zone interdite jusqu'à ce qu'elle ne puisse plus être élargie. Pour cela, on définit un CSP dynamique où, chaque fois qu'une solution est trouvée, une nouvelle contrainte conduisant à réduire l'espace de recherche est postée dynamiquement. Ce processus s'arrête lorsque la zone interdite ne peut plus être élargie. La section 6.3.2 fournit un exemple détaillé. La mise en œuvre de notre approche a été effectuée en *Gecode* (cf. l'annexe A.1). Nous réutilisons les résultats obtenus sur les représentations condensées de motifs et les skypatterns [Soulet *et al.*, 2011]

afin de générer moins de motifs candidats à être des solutions tout en garantissant une approche correcte et complète. Concrètement, cela signifie que nous générons uniquement les motifs de la représentation condensée par rapport à M' où M' est l'ensemble des mesures skylineables de M (cf. le principe, provenant d'AETHERIS, des mesures skylineables est explicité à la section 6.1.2).

Enfin, la déclarativité du cadre PPC nous permet facilement de gérer les contraintes offrant plusieurs sortes de souplesse et conduit à un cadre unifié pour l'extraction des skypatterns souples.

La section 6.3.1 décrit comment l'extraction des skypatterns peut être effectuée en utilisant les CSP dynamiques. La section 6.3.2 fournit un exemple. La section 6.3.3 montre que les skypatterns souples peuvent être extraits de la même manière que les skypatterns (durs). Enfin, la section 6.3.4 est consacrée aux contraintes de fermeture.

6.3.1 Extraction des skypatterns

Soit M un ensemble de mesures, et $\mathtt{x}$ la variable représentant le motif (inconnu) recherché. Les modifications des CSP sont effectuées uniquement par l'ajout de nouvelles contraintes (cf. la section 2.6 à la page 42).

Nous définissons la séquence $P_1, P_2, \ldots, P_n$ de CSP où chaque $P_i = (\{\mathtt{x}\}, \mathcal{L}_\mathcal{I}, \mathtt{q}_i(\mathtt{x}))$ et :
— $\mathtt{q}_1(\mathtt{x}) = \mathtt{fermé}_{M'}(\mathtt{x})$
— $\mathtt{q}_{i+1}(\mathtt{x}) = \mathtt{q}_i(\mathtt{x}) \wedge \phi(s_i, \mathtt{x})$ où s_i est la première solution à la requête $\mathtt{q}_i(\mathtt{x})$.
La contrainte $\mathtt{fermé}_{M'}(\mathtt{x})$ stipule que $\mathtt{x}$ doit être un motif fermé par rapport à M'. Ce choix permettra de réduire le nombre de motifs redondants [29].

La contrainte $\phi(s_i, \mathtt{x}) \equiv (s_i \not\succ_M \mathtt{x})$ indique que la solution suivante recherchée ne sera pas dominée par s_i. En utilisant une courte preuve par induction, on en déduit que la requête $\mathtt{q}_{i+1}(\mathtt{x})$ recherchera un motif $\mathtt{x}$ qui ne sera dominé par aucun des motifs $s_1, s_2, \ldots, s_i$.

Chaque fois que la première solution s_i de la requête $\mathtt{q}_i(\mathtt{x})$ est trouvée, une nouvelle contrainte $\phi(s_i, \mathtt{x})$, fondée sur les valeurs des mesures pour s_i, est postée dynamiquement. Ce processus s'arrête lorsqu'il existe n tel que la requête $\mathtt{q}_{n+1}(\mathtt{x})$ n'a pas de solution.

Compte-tenu de la définition de la Pareto dominance (cf. la définition 6.1), $(s_i \succ_M \mathtt{x})$ se formule par :

$$(s_i \succ_M \mathtt{x}) \equiv \left(\bigwedge_{m \in M} m(s_i) \geq m(\mathtt{x}) \right) \wedge \left(\bigvee_{m \in M} m(s_i) > m(\mathtt{x}) \right)$$

Pour les skypatterns, $\phi(s_i, \mathtt{x})$ stipule que $(s_i \not\succ_M \mathtt{x})$, d'où :

$$\phi(s_i, \mathtt{x}) \equiv \left(\bigvee_{m \in M} m(s_i) < m(\mathtt{x}) \right) \vee \left(\bigwedge_{m \in M} m(s_i) = m(\mathtt{x}) \right)$$

Mais, les n motifs extraits $s_1, s_2, \ldots, s_n$ ne sont pas nécessairement tous des skypatterns. Certains d'entre eux ne peuvent être que des motifs "intermédiaires" utilisés tout simplement pour agrandir la zone interdite. Une étape de post-traitement est donc nécessaire afin d'éliminer tous les candidats qui ne seraient pas des skypatterns. Un candidat s_i n'est pas un skypattern si et seulement si il existe s_j $(1 \leq i < j \leq n)$ tel que $s_j \succ_M s_i$.

L'extraction des skypatterns s'effectue en deux étapes :

29. La contrainte de fermeture est utilisée pour réduire la redondance entre motifs. En effet, les **skypatterns fermés** constituent une représentation condensée exacte de l'ensemble des skypatterns [Soulet *et al.*, 2011]

Trans.	Items					
t_1	A	B	C	D	E	F
t_2	A	B	C	D	E	F
t_3	A	B				
t_4				D		
t_5	A		C			
t_6					E	

Item	Prix
A	10
B	55
C	70
D	30
E	15
F	25

TABLE 6.2 – Jeu de données jouet.

1. Calculer l'ensemble $S = \{s_1, s_2, \ldots, s_n\}$ des candidats à l'aide de CSP dynamiques.
2. Filtrer tous les motifs $s_i \in S$ qui ne sont pas skypatterns.

Le nombre de candidats (n) reste de taille raisonnable en pratique pour les expérimentations que nous avons menées (cf. la figure 6.7 et le tableau 6.3 pour les jeux de données de l'*UCI*, et le tableau 6.6 pour l'étude de cas en chémoinformatique).

6.3.2 Exemple

Cette sous-section illustre le calcul des skypatterns à l'aide des CSP dynamiques (cf. la section 6.3.1) en considèrant un jeu de données jouet. Elle montre tout particulièrement comment la zone interdite est successivement agrandie au fur et à mesure de la résolution. Le jeu de données jouet est décrit au tableau 6.2, et $M = \{\texttt{freq}, \texttt{aire}\}$ est l'ensemble des mesures considérées, ainsi $M' = \{\texttt{freq}\}$.

Soit P_1 le CSP dynamique initial (cf. la section 6.3.1). $P_1 = (\{\texttt{x}\}, \mathcal{L}_{\mathcal{I}}, \texttt{q}_1(\texttt{x}))$ et la requête $\texttt{q}_1(\texttt{x})$ est définie par : $\texttt{q}_1(\texttt{x}) = \texttt{fermé}_{M'}(\texttt{x})$. La première solution de cette requête est le motif (fermé) s_1=$ABCDEF$, pour lequel on a $\texttt{freq}(s_1) = 2$ et $\texttt{aire}(s_1) = 12$. La première ébauche de la zone interdite ainsi obtenue est décrite à la figure 6.4a.

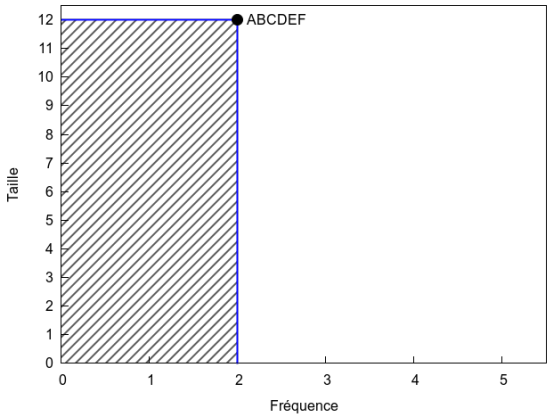
La requête $\texttt{q}_2(\texttt{x}) = \texttt{fermé}_{M'}(\texttt{x}) \wedge (s_1 \not\succ_M \texttt{x})$ impose que le prochain motif recherché soit, certes un motif fermé, mais surtout qu'il ne soit pas dominé par $s_1 = ABCDEF$. La première solution de la requête $\texttt{q}_2(\texttt{x})$ est le motif $s_2 = AB$, pour lequel on a $\texttt{freq}(s_2) = 3$ et $\texttt{aire}(s_2) = 6$. La zone interdite est alors agrandie comme le montre la figure 6.4b.

Puis, la requête $\texttt{q}_3(\texttt{x}) = \texttt{fermé}_{M'}(\texttt{x}) \wedge (s_1 \not\succ_M \texttt{x}) \wedge (s_2 \not\succ_M \texttt{x})$ impose que le prochain motif recherché soit toujours un motif fermé, mais surtout qu'il ne soit dominé ni par s_1, ni par s_2. La première solution de la requête $\texttt{q}_3(\texttt{x})$ est le motif $s_3 = AC$, pour lequel on a $\texttt{freq}(s_3) = 3$ et $\texttt{aire}(s_3) = 6$. La zone interdite peut encore être agrandie, comme le montre la figure 6.4c.

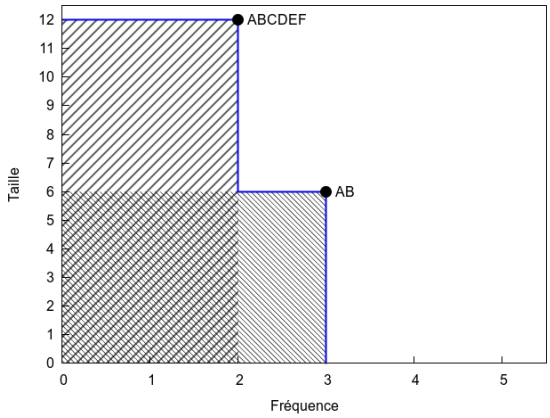
La requête suivante est $\texttt{q}_4(\texttt{x}) = \texttt{q}_3(\texttt{x}) \wedge (s_3 \not\succ_M \texttt{x})$, où $\texttt{x}$ ne doit être dominé ni par s_1, ni par s_2, ni par s_3. La première solution de $\texttt{q}_4(\texttt{x})$ est le motif $s_4 = A$. La zone interdite est encore agrandie comme le montre la figure 6.4d.

Enfin, la requête $\texttt{q}_5(\texttt{x}) = \texttt{q}_4(\texttt{x}) \wedge (s_5 \not\succ_M \texttt{x})$ ne possède aucune solution. La résolution s'achève donc, et la zone interdite courante devient la zone interdite finale.

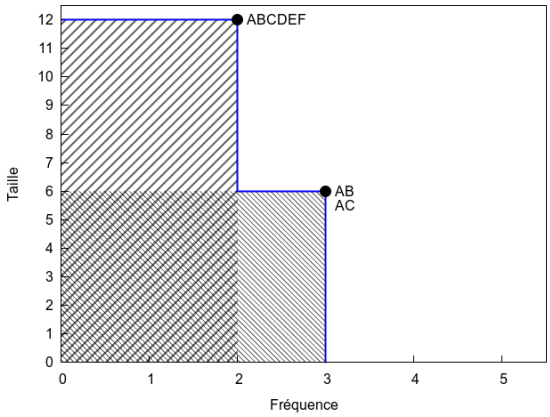
Il faut noter que, sur cet exemple jouet, tous les candidats générés (i.e. les solutions s_i) sont des skypatterns. Les expérimentations menées aux sections 6.4 et 6.5 traitent différents exemples où tous les candidats ne seront pas nécessairement des skypatterns.



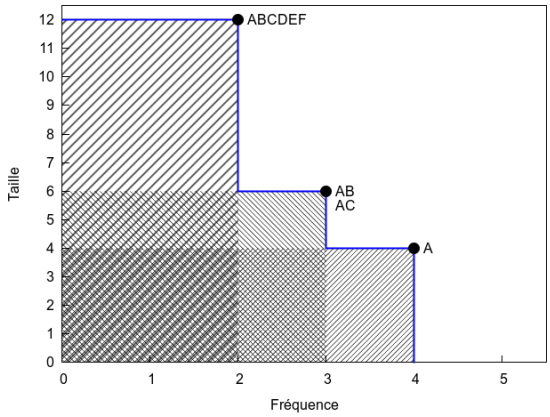
(a) 1ère solution : $s_1 = ABCDEF$.



(b) 2nde solution : $s_2 = AB$.



(c) 3ème solution : $s_3 = AC$.



(d) Dernière solution : $s_4 = A$.

FIGURE 6.4 – Résolution de l'exemple jouet à l'aides des CSP dynamiques.

6.3.3 Extraction des skypatterns souples

Les skypatterns souples sont extraits de manière analogue aux skypatterns (durs) en utilisant les CSP dynamiques (cf. la section 6.3.1). Chaque type de skypatterns souples possède sa propre contrainte $\phi(s_i, \mathtt{x})$ associée à sa relation de dominance.

Pour les edge-skypatterns, $\phi(s_i, \mathtt{x})$ stipule que $(s_i \not\gg_M \mathtt{x})$ (cf. la définition 6.3) :

$$\phi(s_i, \mathtt{x}) \equiv \bigvee_{m \in M} m(s_i) \leq m(\mathtt{x})$$

Pour les δ-skypatterns, $\phi(s_i, \mathtt{x})$ stipule que $(s_i \not\succ^{\delta}_{M} \mathtt{x})$ (cf. la définition 6.5) :

$$\phi(s_i, \mathtt{x}) \equiv \bigvee_{m \in M} (1 - \delta) \times m(s_i) \leq m(\mathtt{x})$$

Les n motifs extraits $s_1, s_2, \ldots, s_n$ ne sont pas nécessairement tous des skypatterns souples. Certains d'entre eux peuvent être des motifs "intermédiaires" permettant uniquement d'agrandir la zone interdite.

De manière analogue aux skypatterns (durs), un post-traitement est donc nécessaire afin d'éliminer tous les candidats qui ne seraient pas des skypatterns souples.

De manière analogue aux skypatterns (durs), l'extraction des skypatterns souples s'effectue elle aussi en deux étapes :

1. Calculer l'ensemble $S = \{s_1, s_2, \ldots, s_n\}$ des candidats à l'aide des CSP dynamiques.
2. Filtrer tous les motifs $s_i \in S$ qui ne sont pas des skypatterns souples.

Là encore, le nombre de candidats (n) reste de taille raisonnable dans la pratique pour les expérimentations que nous avons menées (cf. le tableau 6.3 pour les jeux de données de l'*UCI*, et le tableau 6.6 pour toxicophores).

6.3.4 Contraintes de fermeture

Cette section est consacrée à l'encodage booléen des contraintes de fermeture de la forme $\texttt{fermé}_{M'}(\texttt{x})$ qui impose qu'un motif $\texttt{x}$ soit fermé par rapport à un ensemble de mesures M' Un premier encodage de la contrainte $\texttt{fermé}_{M'}(\texttt{x})$ où $M' = \{\texttt{freq}\}$ a été proposé à la section 3.2.2 (cf. l'équation (3.5)).

Soit $\mathcal{I}$ un ensemble de n items, $\mathcal{T}$ un ensemble de m transactions, et $\mathbf{d} = (d_{t,i})_{t\in\mathcal{T}, i\in\mathcal{I}}$ la matrice 0/1 associée et définie par : $\forall t \in \mathcal{T}, \forall i \in \mathcal{I}, (d_{t,i} = 1)$ si et seulement si $(i \in t)$.

Comme nous l'avons présenté à la section 3.2, l'encodage booléen d'un motif inconnu $\texttt{x}$ s'effectue à l'aide de :
— n variables booléennes $\{X_1, X_2, \ldots, X_n\}$ tq : $\forall i \in \mathcal{I},\ (X_i = 1)$ si et seulement si $(i \in \texttt{x})$, et
— m variables booléennes $\{T_1, T_2, \ldots, T_m\}$ tq : $\forall t \in \mathcal{T},\ (T_t = 1)$ si et seulement si $(\texttt{x} \subseteq t)$
La relation entre le motif recherché $\texttt{x}$ et le jeu de données $\boldsymbol{r}$ (i.e. $\mathcal{I}$ et $\mathcal{T}$) considéré est établie via les contraintes réifiées de l'équation (3.2) rappelée ci-dessous :

$$\forall t \in \mathcal{T}, (T_t = 1) \iff \sum_{i\in\mathcal{I}} X_i \times (1 - d_{t,i}) = 0$$

Soit $M' = \{\texttt{min}\}$ et val une fonction qui associe une valeur d'attribut à chaque item. Si un item i appartient au motif $\texttt{x}$, alors sa valeur $val(i)$ doit être supérieure ou égale à la valeur minimale. Réciproquement, si cette valeur est supérieure ou égale à la valeur minimale, alors l'item i doit appartenir au motif $\texttt{x}$ (sinon, $\texttt{x}$ ne pourrait pas être maximal pour l'inclusion). Ainsi, l'encodage booléen de la contrainte $\texttt{fermé}_{M'}(\texttt{x})$ où $M' = \{min\}$ est le suivant :

$$\forall i \in \mathcal{I}, (X_i = 1) \iff val(i) \geq \min_{j\in\texttt{x}} \{val(j)\} \tag{6.5}$$

Soit $M' = \{\texttt{freq}\}$, la contrainte $\texttt{fermé}_{M'}(\texttt{x})$ impose que le motif $\texttt{x}$ n'ait pas de sur-ensemble de même fréquence. Son encodage booléen est réalisé par l'équation (3.2) et l'équation (3.5) (rappelée ci-dessous) :

$$\forall i \in \mathcal{I}, (X_i = 1) \iff \sum_{t\in\boldsymbol{r}} T_t \times (1 - d_{t,i}) = 0$$

6.4 Expérimentations sur des jeux de données de l'*UCI*

Les expérimentations menées portent sur 23 jeux de donnés de l'*UCI*. Ces jeux de données ont été choisis en raison de leur diversité en termes de taille et de densité (cf. les colonnes 2, 3 et 4 du tableau 6.3).

Nous avons considéré l'ensemble de mesures suivant : $M = \{\texttt{freq}, \texttt{max}, \texttt{aire}, \texttt{mean}, \texttt{émergence}_i\}$. Les skypatterns sont extraits suivant 6 ensembles de mesures : $M = M_6$ et les 5 sous-ensembles de 4 mesures issus de M, notés M_1,M_2,M_3,M_4,M_5. Pour `mean` (cf. le tableau 1.3 à la page 16), des valeurs d'attributs ont été générées aléatoirement dans l'intervalle [0..1].

Plusieurs méthodes sont comparées :
— deux méthodes pour le calcul des skypatterns :
1. CP+Sky, qui calcule l'ensemble des candidats à l'aide des CSP dynamiques, puis applique l'opérateur *Sky* pour filtrer les candidats qui ne sont pas des skypatterns,
2. Aetheris, qui calcule l'ensemble de motifs fermés par rapport à M', puis applique l'opérateur *Sky* selon M pour filtrer les motifs qui ne sont pas des skypatterns (cf. la section 6.1.2).

— trois méthodes pour le calcul des skypatterns souples :
1. CP+Edge-Sky, qui calcule l'ensemble des candidats à l'aide des CSP dynamiques, puis applique l'opérateur *Edge-Sky* pour filtrer les candidats qui ne sont pas edge-skypatterns,
2. CP+δ-Sky, qui calcule l'ensemble des candidats à l'aide des CSP dynamiques, puis applique l'opérateur δ-*Sky* pour filtrer les candidats qui ne sont pas δ-skypatterns,
3. Edge-Aetheris, qui est une extension d'Aetheris permettant d'extraire les edge-skypatterns par rapport à un ensemble de mesures M.

Comme Aetheris (décrit à la section 6.1.2 à la page 87), Edge-Aetheris procède en deux étapes : tout d'abord, extraction de la représentation condensée constituée de l'ensemble des motifs fermés par rapport à M' (où M' est l'ensemble de mesures skylineables de M) ; puis, l'opérateur *Edge-Sky* (cf. définition 6.4 à la page 89) est appliqué selon M. La première étape est la même que pour Aetheris, tandis que la seconde a été réalisée en mettant en œuvre l'opérateur *Edge-Sky* qui utilise la relation de dominance stricte.

Pour toutes les méthodes comparées, les temps de calcul reportés comprennent les deux étapes. La seconde étape (i.e. l'étape de filtrage) est mise en œuvre grâce à l'algorithme BNL (cf. section 4.2.1). Toutes les expérimentations ont été réalisées sur un ordinateur exécutant le système d'exploitation Linux avec un processeur Core i3 à 2.13 GHz et une mémoire vive de 4 Go. La mise en œuvre d'Aetheris se réfère à [Soulet *et al.*, 2011]. La mise en œuvre des méthodes fondées sur les CSP dynamiques a été réalisée en *Gecode* (cf. l'annexe A.1).

6.4.1 Extraction des skypatterns

Le tableau 6.3 compare les résultats de CP+Sky avec ceux d'Aetheris sur les 23 jeux de données de l'*UCI* que nous avons retenus. Pour chaque jeu de données et chaque ensemble de mesures [30], nous reportons :
— le nombre de skypatterns [31],
— pour CP+Sky, le nombre de candidats et le temps de calcul associé,
— pour Aetheris, le nombre de motifs fermés et le temps de calcul associé.
Le tableau 6.3 montre que CP+Sky et Aetheris ont des performances comparables. Sur 16 jeux de données, les temps de calcul des deux méthodes sont très faibles et similaires (moins de 30 secondes), quel que soit l'ensemble de mesures. Dans ce qui suit, nous nous focalisons uniquement sur les 7 jeux de données restants.

30. Les valeurs indiquées dans les colonnes (5-9) sont associées à M_6, alors que les valeurs associées aux colonnes (10-14) représentent des valeurs moyennes sur les 5 sous-ensembles de 4 mesures issus de M : M_1, M_2, M_3, M_4 et M_5.
31. C'est évidemment le même pour les deux méthodes.

Jeu de données				$M = \{\texttt{freq}, \texttt{max}, \texttt{aire}, \texttt{mean}, \texttt{émergence}_i\}$					Moyenne sur $\{M_1,M_2,M_3,M_4,M_5\}$				
					CP+Sky		Aetheris			CP+Sky		Aetheris	
	Nombre d'items	Nombre de transactions	densité	Nombre de Skypatterns	Nombre de Candidats	Temps de calcul (sec)	Nombre de motifs fermés	Temps de calcul (sec)	Nombre de Skypatterns	Nombre de Candidats	Temps de calcul (sec)	Nombre de motifs fermés	Temps de calcul (sec)
abalone	28	4,177	0.321	76	5,255	25	9,947	**1**	43.20	3,393.20	6	9,808.80	**1**
anneal	68	798	0.195	187	13,903	12	35,152	**3**	80.60	6,748.20	6	30,606.00	**1**
austral	55	690	0.272	172	49,379	24	243,156	**20**	77.00	19,626.80	18	228,115.20	**13**
breast	43	286	0.231	38	2,311	**1**	7,721	**1**	23.60	1,374.20	**1**	7,369.20	**1**
cleve	43	303	0.325	97	19,370	**8**	77,203	**8**	51.80	11,215.80	5	74,670.80	**4**
cmc	28	1,473	0.357	62	12,760	12	25,649	**1**	36.00	7,702.80	9	25,408.60	**1**
crx	59	690	0.269	143	78,327	**44**	349,721	55	59.40	31,482.80	**19**	317,090.20	27
german	76	1,000	0.276	308	347,957	**426**	3,662,911	652	121.80	148,255.40	**249**	3,525,072.40	305
glass	34	214	0.295	52	2,633	**1**	7,165	**1**	31.80	1,587.80	**1**	6,920.80	**1**
heart	38	270	0.368	154	16,960	**6**	72,618	8	73.60	9,118.00	**4**	70,124.20	**4**
hepatic	45	155	0.421	237	41,096	**10**	222,333	31	103.20	16,156.20	**4**	206,742.00	13
horse	75	300	0.235	63	28,275	**27**	191,177	47	34.20	12,609.60	**16**	182,982.20	23
hypo	47	3,163	0.389	478	221,032	**469**	1,604,864	893	204.80	145,359.40	**303**	1,565,797.20	481
iris	15	150	0.333	7	86	**1**	93	**1**	5.80	72.40	**1**	91.80	**1**
lymph	59	142	0.322	161	26,200	**7**	116,030	26	64.00	12,532.40	**4**	105,260.00	11
mushroom	119	8,124	0.193	176	14,599	1,186	1,153,229	**548**	75.40	5,767.60	1,137	995,808.40	**221**
new-thyroid	21	215	0.287	15	200	**1**	288	**1**	12.00	151.80	**1**	285.00	**1**
page	35	941	0.314	92	4,482	8	21,121	**2**	49.00	2,576.00	6	20,207.80	**1**
pima	26	768	0.346	53	1,400	3	12,559	**1**	32.40	943.80	3	12,439.40	**1**
tic-tac-toe	29	258	0.344	82	11,206	11	43,318	**3**	51.20	7,867.40	9	42,902.20	**2**
vehicle	58	846	0.327	280	164,152	172	745,353	**138**	126.20	73,390.60	**69**	689,937.80	84
wine	45	178	0.311	43	7,780	**2**	36,671	4	25.40	4,538.60	**2**	34,397.00	**2**
zoo	43	101	0.394	56	4,654	**1**	14,431	**1**	34.20	2,642.60	**1**	12,851.20	**1**

Table 6.3 – Comparaison entre CP+Sky et Aetheris sur 23 jeux de données de l'*UCI*.

La figure 6.5 montre le nuage de points des temps de calcul de CP+Sky et Aetheris pour les 7 jeux de données nécessitant plus de 30s de calcul. Chaque point représente l'extraction des skypatterns pour un jeu de données et un ensemble de mesures M_i ($i \in [1..6]$) : sa coordonnée x (échelle logarithmique) correspond au temps de calcul pris par CP+Sky et sa coordonnée y (échelle logarithmique) représente le temps de calcul d'Aetheris. Une couleur différente est associée à chaque jeu de données. La ligne $y = x$ indique les temps de calcul identiques aux deux méthodes. Comme la plupart des points sont au dessus de cette ligne, cela signifie que Aetheris met plus de temps comparé à CP+Sky. En considérons toutes les mesures (i.e. M_6), l'accélération est de 1.9 (resp. 1.53) sur hypo (resp. german). La seule exception est le jeu de données mushroom, qui est le plus grand (en termes de nombre de transactions et de nombre d'items), mais qui est de faible densité (environ 18%). mushroom possède un nombre relativement faible [32] de motifs fermés, ce qui favorise Aetheris car ce dernier tire profit de la concision de la représentation condensée qu'il calcule.

Afin de mieux mesurer l'impact de la densité sur les temps de calcul des deux méthodes CP+Sky et Aetheris (en particulier sur mushroom), nous avons généré plusieurs jeux de données de même taille (en nombre d'items et de transactions) que mushroom et nous avons fait varié la densité de 0.15 à 0.50. La figure 6.6 montre l'évolution du temps de calcul en fonction de la densité pour les deux méthodes. Comme les items sont générés de façon aléatoire pour chaque valeur de densité, le nombre de motifs fermés augmente en fonction de la densité (jusqu'à environ 50%) et diminue après (jusqu'à avoir un seul motif à 100%). Ainsi, le temps de calcul d'Aetheris augmente également en fonction de ce paramètre. CP+Sky présente un comportement bien plus complexe en raison de l'élagage dynamique. En effet, pour des valeurs

32. Au regard de sa taille

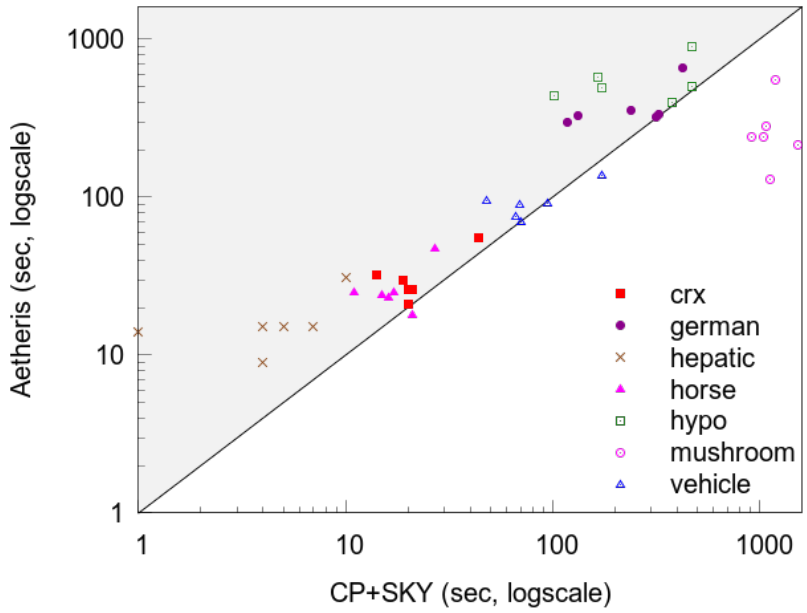


FIGURE 6.5 – Comparaison des temps de calcul sur les 7 jeux de données sélectionnés pour M_i, $i \in [1..6]$.

de densité ≤ 0.38, AETHERIS surclasse clairement CP+SKY. Toutefois, ce comprtement s'inverse pour une densité 0.39. Ce comportement peut problablement s'expliquer par le fait que les performances de CP+SKY dépendent du nombre de candidats générés mais aussi du temps passé pour obtenir un candidat. Toutefois, l'augmentation du nombre de candidats peut être bénéfique si ces candidats permettent d'élaguer de manière significative l'espace de recherche. L'étude expérimentale montre toutefois un bon compromis avec une densité proche de 0.5.

Un autre résultat intéressant fourni par le tableau 6.3 est le nombre de motifs fermés obtenus par AETHERIS par rapport aux nombre de candidats générés par CP+SKY, c'est à dire le nombre de *motifs représentatifs* pour les deux méthodes. Le nombre de candidats générés par CP+SKY demeure faible (milliers) par rapport au nombre important de motifs fermés (millions) produits par AETHERIS. D'autre part, nous pouvons remarquer que le nombre de skypatterns est peu élevé (centaines). La figure 6.7 résume ces résultats sous forme graphique. Elle reporte, pour chaque ensemble de mesures M_i ($i \in [1..6]$), le nombre de motifs fermés (AETHERIS) et le nombre de candidats (CP+SKY) obtenus sur les 7 jeux de données étudiés à la figure 6.5. Très clairement, le nombre de motifs représentatifs requis par CP+SKY reste moins élevé comparé à AETHERIS.

6.4.2 Extraction des skypatterns souples

Cette section montre la faisabilité de l'extraction des skypatterns souples sur les jeux de données de l'*UCI*. Pour les δ-skypatterns, δ a été fixé à $\{0.05, 0.10, 0.15, 0.20\}$.

6.4.2a Extraction des Edge-skypatterns

Le tableau 6.4 compare les résultats de CP+EDGE-SKY avec ceux de EDGE-AETHERIS sur les 23 jeux de données de l'*UCI*. Pour chaque jeu de données et chaque ensemble de mesures, nous reportons :

— le nombre d'edge-skypatterns qui ne sont pas des skypatterns (durs),

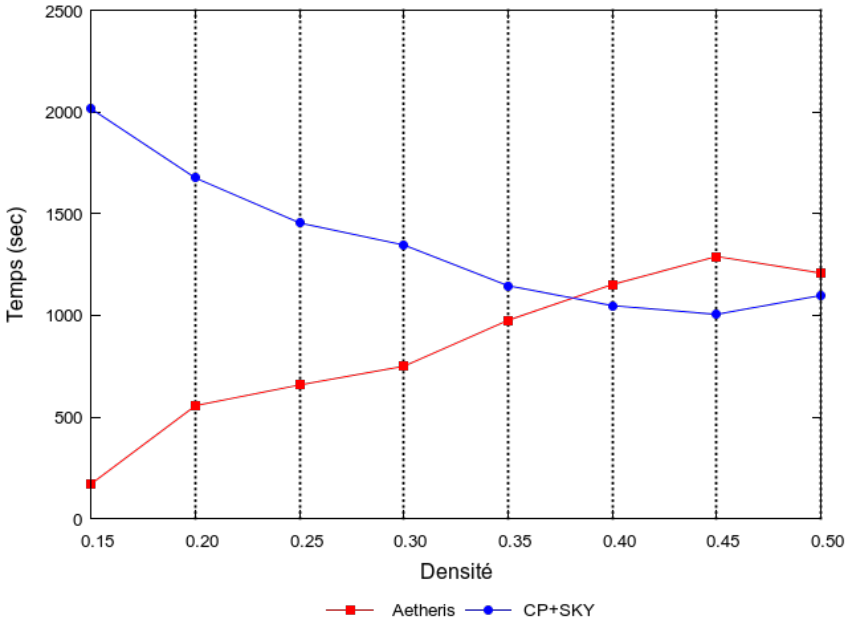


FIGURE 6.6 – Impact de la densité sur les temps de calcul de CP+SKY et AETHERIS.

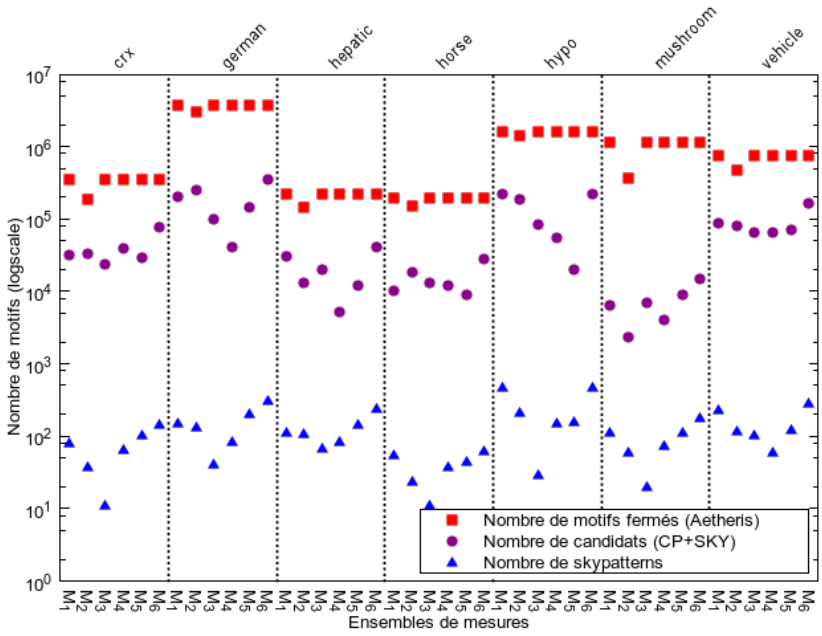


FIGURE 6.7 – Comparaison du nombre de fermés, de candidats et de skypatterns sur 7 jeux de données de l'*UCI*.

— pour CP+EDGE-SKY, le nombre de candidats et le temps de calcul associé,
— pour EDGE-AETHERIS, le nombre de motifs fermés et le temps de calcul associé.

Comme l'indiquent les résultats du tableau 6.4, les deux méthodes CP+EDGE-SKY et EDGE-AETHERIS obtiennent des résultats comparables en temps de calcul sur la plupart des jeux de données. La figure 6.8

Jeu de données	$M = \{\texttt{freq}, \texttt{max}, \texttt{aire}, \texttt{mean}, \texttt{émergence}_1\}$					Moyenne sur $\{M_1,M_2,M_3,M_4,M_5\}$				
		CP+Edge-Sky		Edge-Aetheris			CP+Edge-Sky		Edge-Aetheris	
	Nombre d'edge-skypatterns	Nombre de candidats	Temps de calcul (sec)	Nombre de motifs fermés	Temps de calcul (sec)	Nombre d'edge-skypatterns	Nombre de candidats	Temps de calcul (sec)	Nombre de motifs fermés	Temps de calcul (sec)
abalone	2,634	3,815	36	9,947	**25**	2,380.40	3,220.40	30	9,808.80	**21**
anneal	3,162	10,375	**28**	35,152	32	2,298.80	6,642.60	25	30,606.00	**14**
austral	11,714	18,147	69	243,156	**44**	14,997.40	24,879.00	74	228,115.20	41
breast	1,409	2,845	**1**	7,721	**1**	2,024.20	3,938.80	**1**	7,369.20	**1**
cleve	14,636	27,142	19	77,203	**15**	18,813.40	27,795.60	15	74,670.80	**10**
cmc	14,406	18,866	31	25,649	**21**	13,431.20	19,227.80	28	25,408.60	**17**
crx	29,068	41,879	134	349,721	104	32,216.60	61,396.00	94	317,090.20	**64**
german	93,097	499,848	**1,157**	3,662,911	3,100	89,001.50	503,624.00	**1,779**	3,525,072.40	4,810
glass	2,296	4,729	**1**	7,165	**1**	3,003.60	5,158.20	**1**	6,920.80	**1**
heart	15,563	52,074	15	72,618	**12**	18,640.60	42,207.60	17	70,124.20	**8**
hepatic	15,002	27,182	24	222,333	**20**	17,195.80	26,937.80	25	206,742.00	**12**
horse	13,068	22,437	54	191,177	**45**	88,970.00	79,923.80	67	182,982.20	**56**
hypo	278,625	1,399,848	**1,343**	1,604,864	3,331	252,248.50	1,003,624.00	**1,416**	1,565,797.20	2,838
iris	55	65	**1**	93	**1**	51.40	69.80	**1**	91.80	**1**
lymph	8,286	15,271	**19**	116,030	34	7,201.40	8,470.00	**18**	105,260.00	20
mushroom	21,639	49,848	3,241	1,153,229	**1,225**	28,784.00	53,624.00	3,445	995,808.40	**938**
new-thyroid	119	201	**1**	288	**1**	145.20	235.40	**1**	285.00	**1**
page	2,675	12,113	18	21,121	**10**	2,926.40	15,032.20	14	20,207.80	**7**
pima	1,778	11,781	5	12,559	**1**	2,377.00	19,162.60	3	12,439.40	**1**
tic-tac-toe	6,800	19,757	16	43,318	**22**	7,113.80	23,371.00	19	42,902.20	**25**
vehicle	76,732	146,151	687	745,353	399	82,732.60	162,729.00	776	689,937.80	**424**
wine	3,155	5,507	5	36,671	**2**	4,430.40	6,262.60	6	34,397.00	**3**
zoo	2,254	8,427	2	14,431	**1**	3,208.80	7,888.40	**1**	12,851.20	**1**

Table 6.4 – Comparaison entre CP+Edge-Sky et Edge-Aetheris sur 23 jeux de données de l'*UCI*.

confirme cette tendance pour chaque ensemble de mesures M_i ($i \in [1..6]$), sur les 7 jeux de données nécessitant plus de $30s$ de calcul. En effet, presque tous les points sont proches de la droite $y = x$. Notons toutefois, que sur les deux jeux de données hypo et german, CP+Edge-Sky devance largement Edge-Aetheris. Pour M_6, l'accélération est de 2.67 (resp. 2.48) sur hypo (resp. german) (cf. le tableau 6.4). Une fois de plus, la seule exception est le jeu de données mushroom, pour lequel Edge-Aetheris surpasse clairement CP+Edge-Sky (Edge-Aetheris est environ 2.64 plus rapide que CP+Edge-Sky) car mushroom a une densité faible (environ 19%), ce qui se traduit par un petit nombre de motifs fermés par rapport à sa taille (e.g. german est plus petit en termes de nombre d'items et de nombre de transactions mais a trois fois plus de motifs fermés) ce qui favorise Edge-Aetheris fondée sur l'extraction de motifs fermés.

Enfin, le nombre de candidats générés par CP+Edge-Sky demeure faible par rapport au nombre de motifs fermés produits par Edge-Aetheris.

6.4.2b Extraction des δ-skypatterns

Le tableau 6.5 indique, pour chaque jeu de données de l'*UCI* et chaque valeur de $\delta \in \{0.05, 0.10, 0.15, 0.20\}$, le nombre de δ-skypatterns qui ne sont pas des edge-skypatterns, le nombre de candidats et le temps de calcul nécessaire. Les résultats sont donnés pour l'ensemble des mesures (i.e. M_6). Même si le nombre de δ-skypatterns augmente avec δ, notre approche reste efficace : il y a seulement 8 expérimentations sur 115 nécessitant plus de $3,000$ secondes.

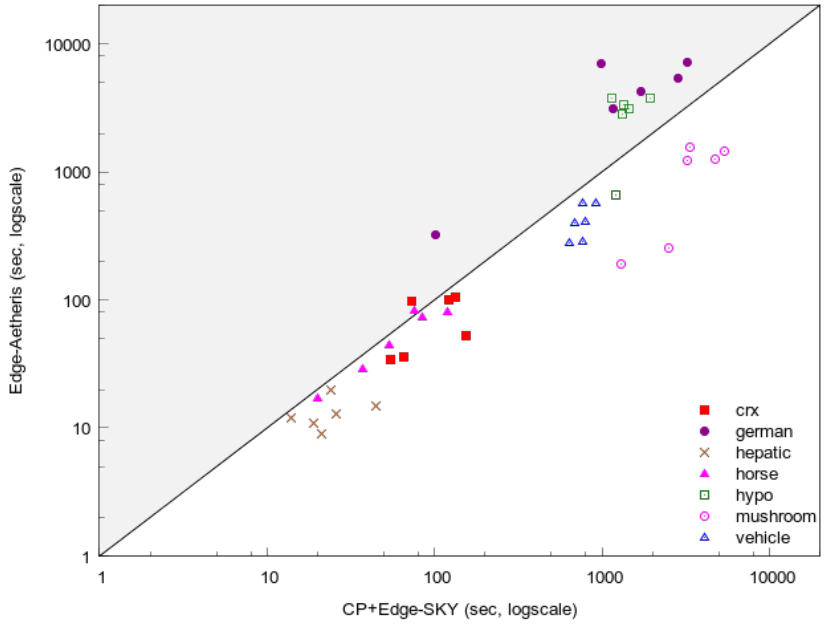


FIGURE 6.8 – Comparaison des temps de calcul sur les 7 jeux de données de l'*UCI* pour M_i, $i \in [1..6]$ (edge-skypatterns).

				CP+δ-SKY ($\delta = 5\%$)		CP+δ-SKY ($\delta = 10\%$)		CP+δ-SKY ($\delta = 15\%$)		CP+δ-SKY ($\delta = 20\%$)	
Jeu de données	Nombre d'items	Nombre de transactions	Densité	Nombre de δ-skypatterns	Temps de calcul (sec)	Nombre de δ-skypatterns	Temps de calcul (sec)	Nombre de δ-skypatterns	Temps de calcul (sec)	Nombre de δ-skypatterns	Temps de calcul (sec)
abalone	28	4,178	0.321	1,373	38	1,432	38	6,303	38	7,256	42
anneal	68	798	0.195	8,184	34	20,242	35	24,029	37	27,214	36
austral	55	690	0.272	34,205	70	68,855	99	69,487	102	70,652	113
breast	43	286	0.231	17	1	1,651	1	2,429	1	2,443	1
cleve	43	303	0.325	4,466	19	30,605	22	30,952	22	50,275	23
cmc	28	1,474	0.357	3,297	32	3,351	32	11,848	33	14,020	33
crx	59	690	0.269	73,627	151	73,707	159	105,344	166	165,782	167
german	76	1,000	0.276	170,169	2,614	230,457	2,995	270,435	3,439	290,654	3,483
glass	34	216	0.295	109	1	1,531	1	2,491	1	4,035	1
heart	38	270	0.368	644	16	34,136	16	42,685	18	44,114	18
hepatic	45	155	0.421	6,122	24	45,572	25	50,686	25	60,857	26
horse	75	300	0.235	39,149	60	43,073	66	55,175	68	74,275	71
hypo	47	3,163	0.389	104,147	1,387	115,126	1,402	116,654	1,463	117,089	1,487
iris	15	151	0.333	20	1	27	1	49	1	67	1
lymph	59	142	0.322	49,846	19	59,753	20	62,143	20	65,946	21
mushroom	119	8,124	0.193	33,757	3,328	99,852	3,336	129,383	3,407	150,965	3,614
new-thyroid	21	216	0.287	41	1	137	1	154	1	173	1
page	35	941	0.314	7,136	19	9,714	19	17,387	21	19,094	22
pima	26	768	0.346	518	5	3,439	5	4,308	5	4,358	6
tic-tac-toe	29	259	0.344	4,078	18	18,584	19	20,130	20	22,576	21
vehicle	58	846	0.327	716	689	2,457	751	3,789	782	4,369	787
wine	45	179	0.311	2,490	5	4,422	5	7,507	5	13,407	6
zoo	43	102	0.394	3,361	2	4,829	2	7,724	2	8,986	2

TABLE 6.5 – Analyse de l'extraction des δ-skypatterns sur les jeux de données de l'*UCI* pour M_6.

6.5 Étude de cas : découverte de toxicophores

Le but de la présente étude, qui s'inscrit dans le cadre d'une collaboration avec le CERMN, est d'étudier l'apport des skypatterns souples pour la découverte de toxicophores (cf. la section 5.4). Les conditions expérimentales sont les mêmes que pour les jeux de données *UCI* décrites en introduction de la section 6.4.

Nous avons utilisé le jeu de données ECB décrit en section 5.6. Nous avons considéré quatre ensembles de mesures : M_1, M_2, M_3 et M_4 (cf. le tableau 6.6). La redondance est réduite par l'utilisation de skypatterns fermés qui sont une représentation condensée exacte de l'ensemble des skypatterns. Pour les δ-skypatterns, nous avons considéré deux valeurs : δ=10% et δ=20%. Les skypatterns (durs) et les skypatterns souples extraits sont constitués de fragments moléculaires. Pour évaluer la présence de toxicophores dans leur description, une analyse d'experts a conduit à l'identification de toxicophores environnementaux bien connus. Quelques exemples sont décrits dans le tableau 6.9.

6.5.1 Analyse de performance

Le tableau 6.6 compare les résultats de CP+Sky avec ceux d'Aetheris pour les quatre ensembles de mesures sur le jeu de données ECB. Pour chaque ensemble de mesures, nous indiquons :
— le nombre de skypatterns,
— pour CP+Sky, le nombre de candidats et le temps de calcul associé,
— pour Aetheris, le nombre de motifs fermés et le temps de calcul associé.

CP+Sky surpasse Aetheris en termes de temps de calcul. En outre, le nombre de candidats générés par CP+Sky reste petit par rapport au nombre de motifs fermés de la représentation condensée calculée par Aetheris. Notre approche PPC permet de réduire considérablement le nombre de candidats.

	Skypatterns				
		CP+Sky		Aetheris	
	Nombre de Skypatterns	Nombre de candidats	Temps de calcul	Nombre de motifs fermés	Temps de calcul
$M_1 = \{\texttt{émergence}_i, \texttt{freq}\}$	**8**	613	**18m34s**	41,887	**19m00s**
$M_2 = \{\texttt{émergence}_i, \texttt{aromaticité}\}$	**5**	140	**15m32s**	53,201	**21m33s**
$M_3 = \{\texttt{freq}, \texttt{aromaticité}\}$	**2**	456	**16m45s**	157,911	**21m16s**
$M_4 = \{\texttt{émergence}_i, \texttt{freq}, \texttt{aromaticité}\}$	**21**	869	**17m49s**	12,126	**21m40s**

Table 6.6 – Comparaison entre CP+Sky et Aetheris sur le jeu de données ECB.

Le tableau 6.7 indique, pour chaque ensemble de mesures :
— le nombre d'edge-skypatterns qui ne sont pas des skypatterns, le nombre de candidats et le temps de calcul nécessaire,
— le nombre de δ-skypatterns qui ne sont pas des edge-skypatterns, le nombre de candidats et le temps de calcul nécessaire.

L'augmentation du nombre de mesures conduit à un plus grand nombre de skypatterns (souples), en particulier pour des valeurs élevées de δ. En fait, un motif domine rarement tous les autres motifs sur l'ensemble complet de mesures. Cependant, dans nos expérimentations, le nombre de skypatterns souples

	Edge-Skypatterns			δ-Skypatterns					
				$\delta = 10\%$			$\delta = 20\%$		
	CP+Edge-Sky			CP+δ-Sky					
	Nombre d'edge-skypatterns	Nombre de candidats	Temps de calcul	Nombre de δ-skypatterns	Nombre de candidats	Temps de calcul	Nombre de δ-skypatterns	Nombre de candidats	Temps de calcul
$M_1 = \{\texttt{émergence}_\text{j}, \texttt{freq}\}$	**24**	1,746	**19m02s**	**25**	4,204	**20m48s**	**87**	6,253	**22m36s**
$M_2 = \{\texttt{émergence}_\text{j}, \texttt{aromaticité}\}$	**76**	688	**17m51s**	**354**	1,678	**18m14s**	**1,670**	2,816	**23m44s**
$M_3 = \{freq, \texttt{aromaticité}\}$	**72**	1,726	**16m50s**	**352**	4,070	**19m43s**	**1,654**	6,699	**22m25s**
$M_4 = \{\texttt{émergence}_\text{j}, \texttt{freq}, \texttt{aromaticité}\}$	**144**	3,021	**20m27s**	**385**	6,048	**23m36s**	**1,724**	8,986	**30m14s**

Table 6.7 – Analyse de l'extraction des skypatterns souples sur le jeu de données ECB.

reste de taille raisonnable. Pour les edge-skypatterns, il y a un maximum de 144 motifs, tandis que pour les δ-skypatterns, il y a un maximum de 1,724 motifs ($\delta = 20\%$). Enfin, en ce qui concerne les temps de calcul, notre approche est très efficace : le calcul des skypatterns souples nécessite moins de 30 minutes.

6.5.2 Analyse qualitative

Cette section analyse qualitativement les skypatterns et les skypatterns souples extraits en évaluant la présence de toxicophores environnementaux bien connus dans leur description (cf. le tableau tableau 6.9). Par ailleurs, afin de mieux analyser la répartition des différents skypaterns souples extraits, nous avons utilisé la méthode de clustering *k-means* [MacQueen, 1967] avec une métrique euclidienne car c'est l'une des méthodes les plus classiques nécessitant uniquement le nombre de clusters comme paramètre.

6.5.2a Messures d'émergence et de fréquence (M_1)

La figure 6.9 illustre de manière graphique la répartition des skypatterns et des skypatterns souples extraits pour M_1 :

— **Skypatterns**. Seulement 8 skypatterns sont extraits, et 3 toxicophores bien connus sont mis en évidence. Deux d'entre eux sont des composés aromatiques, à savoir le chlorobenzène (le motif $\texttt{p}_1$: {`Clc`}) et les cycles phénoliques (motif $\texttt{p}_2$: {`c1(ccccc1)O`}). Le troisième, le groupement organophosphoré (motif $\texttt{p}_3$: {`OP`, `OP=S`}) est un composant apparaissant dans de nombreux pesticides.

— **Skypatterns souples**. Ils confirment les tendances données par les skypatterns : les cycles aromatiques chlorés substitués (motif $\texttt{p}_4$: {`Clc(ccc)c`, `Clcccc`}), et le groupement organophosphoré (motif $\texttt{p}_5$: {`OP(=S)O)`, `COP(=S)O`}) sont détectés par les edge-skypatterns et par les δ-skypatterns. Par ailleurs, plusieurs motifs contenant ces toxicophores sont extraits.

La figure 6.10 montre le partitionnement en clusters des skypatterns, et des skypatterns souples, trouvés par la méthode *k-means*. La solution calculée contient $k = 3$ clusters distincts. Une analyse de ces clusters montre que plusieurs groupements chimiques sont liés à des activités biologiques spécifiques.

1. Le cluster #1 est constitué de motifs ayant un fort taux de croissance et une faible fréquence. Il comprend 2 skypatterns et 23 skypatterns souples : 8 d'entre eux sont des edge-skypatterns

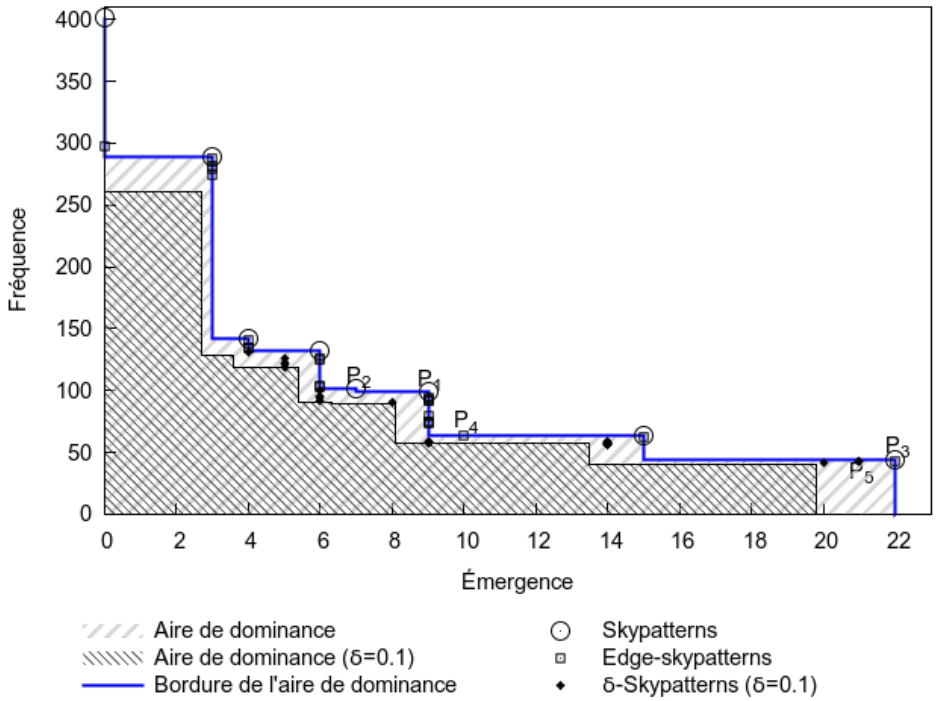


FIGURE 6.9 – Répartition de skypatterns (souples) extraits pour M_1.

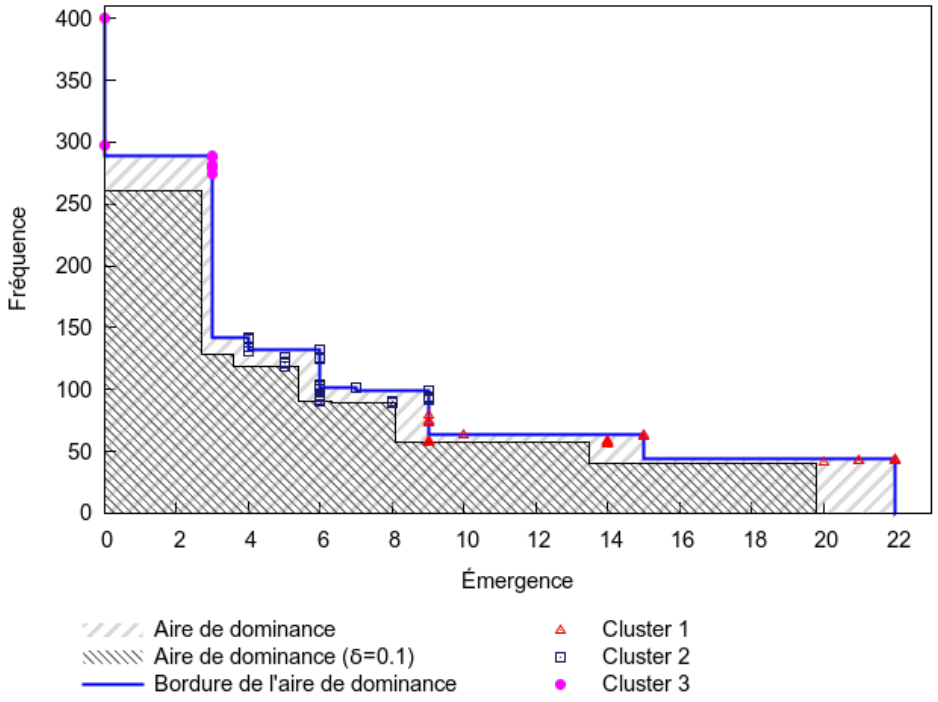


FIGURE 6.10 – Partitionnement des skypatterns (souples) en clusters ($k = 3$) pour M_1.

et 15 sont des δ-skypatterns. D'un point de vue chimique, ce cluster correspond à des composés organophosphorés et à du *benzène substitué par un groupe alkyle* (e.g. {`ccC`, `cccC`}).

2. Le cluster #2 contient 4 skypatterns et 21 skypatterns souples : 11 d'entre eux sont des edge-skypatterns et 10 sont des δ-skypatterns. D'un point de vue chimique, ce cluster met en évidence deux toxicophores bien connus, à savoir le chlorobenzène et les noyaux phénoliques.
3. Le cluster #3 comprend 2 skypatterns et 5 edge-skypatterns. La plupart d'entre eux sont des composés aromatiques, à savoir le noyau benzénique (i.e. avec une fréquence élevée et un faible taux de croissance).

6.5.2b Messures d'émergence et d'aromaticité (M_2)

Les résultats obtenus pour M_2 et M_3 étant similaires, nous présentons uniquement l'analyse qualitative pour M_2. La figure 6.11 montre la répartition des skypatterns et des skypatterns souples extraits pour M_2 :

— les edge-skypatterns conduisent à l'extraction de quatre nouveaux toxicophones :
(i) *composés aromatiques azotés* : l'indole (motif $\mathtt{p}_1$:{`ncc`, `c1ccccc1`}) et le benzoïmidazole (motif $\mathtt{p}_2$: {`ncnc`, `c1ccccc1`}),
(ii) *composés aromatiques sulfurés* : benzothiophène (motif $\mathtt{p}_3$: {`ccs`, `c1ccccc1`}),
(iii) *composés aromatiques oxygénés* : benzofuranne (motif $\mathtt{p}_4$: {`coc`, `c1ccccc1`}),
(iv) *hydrocarbures aromatiques polycycliques* : naphthalène (motif $\mathtt{p}_5$: {`c1ccc2ccccc2cc1`}).

— les δ-skypatterns complètent la liste de composés aromatiques qui n'ont pas été trouvés par les skypatterns, à savoir le biphényle (motif $\mathtt{p}_6$: {`c1ccccc1c2ccccc2`}).

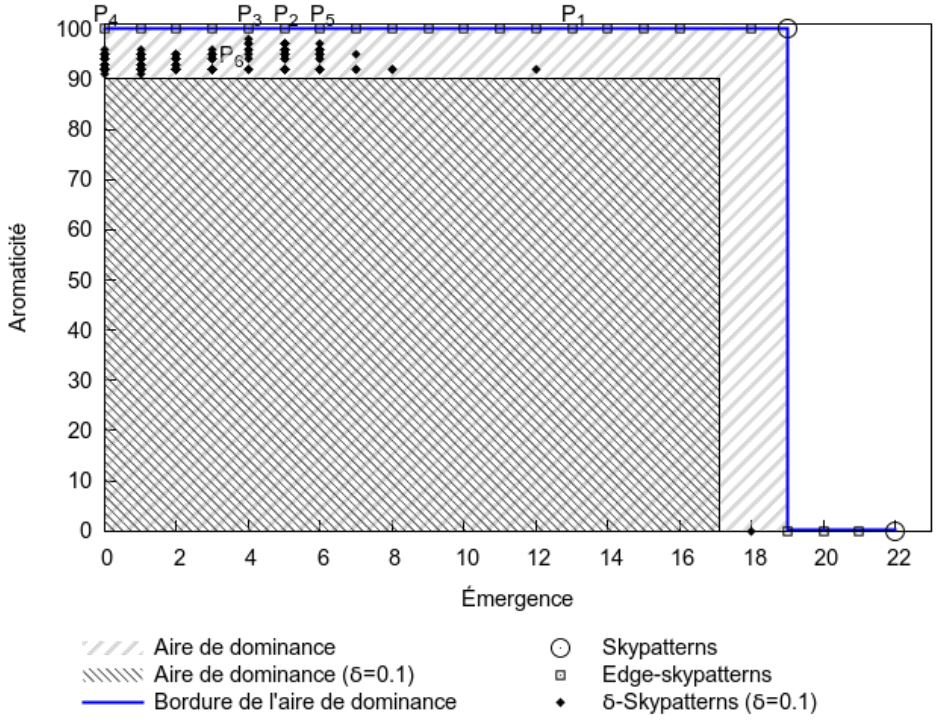


FIGURE 6.11 – Répartition de skypatterns (souples) extraits pour M_2.

La figure 6.12 donne la répartition en clusters des skypatterns et des skypatterns souples trouvés par la méthode *k-means*. Trois ($k = 3$) clusters distincts sont mis en évidence :

1) le cluster #1 est constitué de 3 skypatterns et de 6 edge-skypatterns ayant un très fort taux de croissance et une aromaticité égale à zéro. Ils correspondent à des composés organophosphorés.
2) le cluster #2 contient uniquement des δ-skypatterns. D'un point de vue chimique, il regroupe plusieurs cycles aromatiques différents.
3) le cluster #3 comprend 2 skypatterns et plusieurs edge-skypatterns qui correspondent à des composés aromatiques azotés.

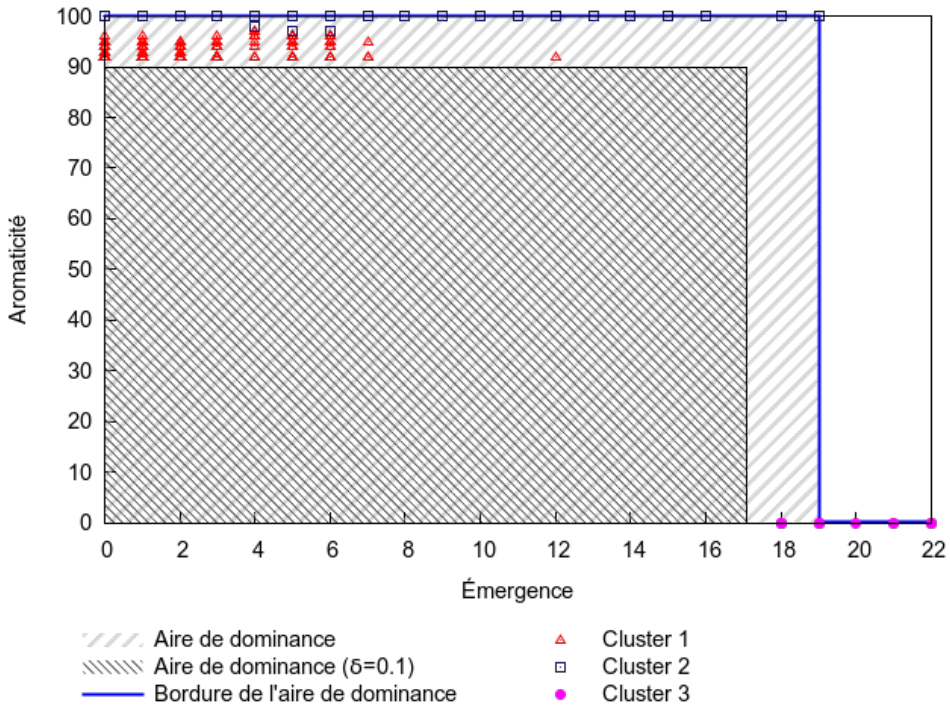


FIGURE 6.12 – Partitionnement des skypatterns (souples) en clusters ($k = 3$) pour M_2.

6.5.2c Messures d'émergence, de fréquence et d'aromaticité (M_4)

Les résultats les plus intéressants sont obtenus en utilisant M_4 (cf. la figure 6.13). Le tableau 6.8 présente l'analyse des ratios pour les skypatterns et skypatterns souples. La colonne 1 indique le nom des toxicophores. Les colonnes $(2-5)$ donnent le nombre de skypatterns et skypatterns souples contenant un fragment complet [33] représentatif de chaque toxicophore ainsi que leurs ratios [34] (entre parenthèses). La colonne 6 (resp. colonne 7) donne le nombre de molécules chimiques (en %) classées H400, i.e., très toxique (resp. H402, i.e., nocif) contenant au moins un fragment représentatif du toxicophore. Les colonnes $(8-10)$ montrent les gains fournis par les skypatterns souples pour la découverte de toxicophores [35]. Les chiffres en gras indiquent un gain supérieur à 1 et la valeur ∞ indique que le toxicophore est détécté uniquement par les skypatterns souples.

21 skypatterns sont extraits (cf. la figure 6.13) et plusieurs toxicophores bien connus sont mis en évidence : le phénol (e_4), les cycles aromatiques chloro-substitués (e_3), le benzène substitué par un groupe

33. Motifs contenant seulement des sous-fragments d'un toxicophore ne sont pas pris en compte.
34. Nombre de skypatterns (souples) contenant ce toxicophore divisé par le nombre total de skypatterns (souples) (cf. en gras à la 2ème ligne).
35. Le gain est le ratio de skypatterns souples divisé par le ratio de skypatterns.

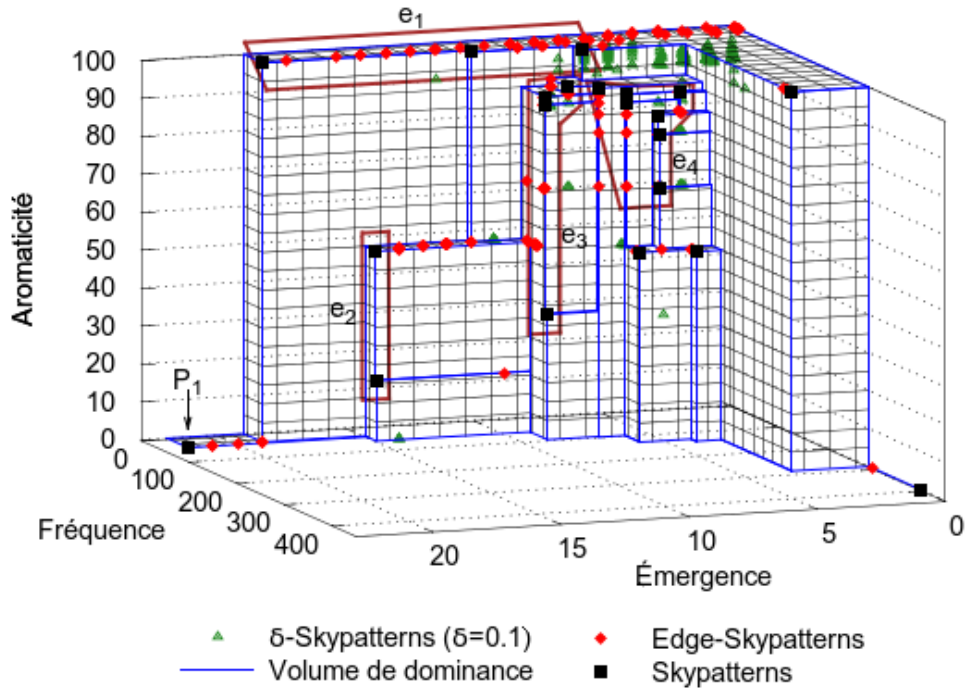


FIGURE 6.13 – Répartition des skypatterns (souples) extraits pour M_4.

Chemical		(2)	(3)	(4)	(5)	(6)	(7)	Gain (3)	Gain (4)	Gain (5)
		21	**165**	**550**	**1889**					
Benzène		4 (0.19)	68 (0.41)	322 (0.59)	1373 (0.73)	63.7	18.9	**2.16**	**3.11**	**3.84**
Chlorobenzène		1 (0.05)	2 (0.01)	51 (0.09)	311 (0.16)	22.5	2.5	0.20	**1.80**	**3.20**
Phénol		1 (0.05)	11 (0.07)	32 (0.06)	302 (0.16)	25.2	3.5	**1.40**	**1.20**	**3.20**
Organophosphorés	**Basique**	2 (0.10)	18 (0.11)	30 (0.05)	40 (0.02)	18.0	2.5	**1.10**	0.50	0.20
	Exotique		38 (0.23)	66 (0.12)	112 (0.06)	18.0	2.5	∞	∞	∞
Cycles aromatiques azotés			15 (0.09)	74 (0.13)	175 (0.09)	8.6	2.0	∞	∞	∞
Cycles aromatiques polycycliques			12 (0.07)	178 (0.32)	302 (0.16)	7.2	3.5	∞	∞	∞
Benzène substitué par un alkyle			4 (0.02)	64 (0.12)	649 (0.34)	30.9	11.7	∞	∞	∞
Aniline				15 (0.03)	259 (0.14)	24.7	11.3		∞	∞
Aniline substitué par un alkyle					157 (0.08)	12.0	7.1			∞
Chlorophénol					168 (0.09)	9.6	1.5			∞
Éthers alkyl-phényliques					106 (0.06)	9.9	3.0			∞
Phénol substitué par un alkyle					61 (0.03)	9.6	1.5			∞
Dichlorobenzène					59 (0.03)	9.9	1.5			∞

(2) Skypatterns
(3) Edge-Skypatterns
(4) δ-Skypatterns ($\delta = 0.1$)
(5) δ-Skypatterns ($\delta = 0.2$)
(6) taux de couverture des molécules H400 (%)
(7) taux de couverture des molécules H402 (%)

TABLE 6.8 – Analyse des ratios des skypattern (souples) extraits pour M_4.

alkyle (e_2), et les composés organophosphorés (motif $\mathtt{p}_1$). En outre, des informations traitant de composés aromatiques azotés sont également extraits (e_1). Le tableau 6.9 détaille la répartition des skypatterns contenant un seul fragment toxicophore complet, en fonction des toxicophores discutés ci-dessus. Comme nous pouvons le constater, très peu de motifs contenant ces toxicophores sont extraits avec les skypatterns durs.

Les skypatterns souples permettent de détecter de manière plus précise les quatre premiers toxicophores (cf. le tableau 6.8). Par exemple, 41% des edge-skypatterns extraits contiennent du benzène, contre 19% pour skypatterns durs (gain de 2.16 : les egde-skypatterns détectent 2.16 fois plus de motifs contenant ce fragment par rapport aux durs). Ce gain atteint environ 3.11 (resp. 3.84) pour $\delta = 0.1$ (resp. 0.2). Ces tendances sont confirmées pour le chlorobenzène et le phénol, où 16% des δ-skypatterns extraits ($\delta = 0.2$) contiennent ces fragments, contre 5% dans le cas dur (gain de 3.20). D'un point de vue chimique, ces fragments couvrent largement les molécules de classe H400 (de 18% à 63.7%) par rapport aux molécules de classe H402 (cf. les colonnes $6 - 7$), confirmant ainsi la nature toxique de ces motifs, en particulier dans le cas souple.

En ce qui concerne les cycles aromatiques décrits précédemment (lignes grises du tableau 6.8), plusieurs nouveaux motifs contenant ces toxicophores sont extraits par les edge-skypatterns. Les δ-skypatterns (avec $\delta = 0.1$) permettent de mieux découvrir ces toxicophores comparé aux edge-skypatterns (gain moyen d'environ 4). En outre, plusieurs motifs contenant de nouveaux fragments d'un grand intérêt sont uniquement détectés par les δ-skypatterns (cf. les lignes jaunes du tableau 6.8), en particulier avec $\delta = 0.2$. 22% de ces motifs contiennent des fonctions amines (14% pour l'aniline et 8% pour les anilines substituées). Ces deux toxicophores, qui couvrent respectivement 24.7% et 12% de molécules classées H400, sont très nocifs pour les organismes aquatiques. Les autres toxicophores sont extraits par les δ-skypatterns avec des ratios allant de 3% à 9%.

Le tableau 6.9 indique, pour chaque toxicophore, la liste des fragments moléculaires extraits (codes smiles) pour M_4 en fonction du type de skypattern. En gras, sont indiqués les fragments complets extraits.

Pour conclure, les skypatterns souples permettent de détecter efficacement des toxicophores bien connus mis en évidence par les skypatterns, et de découvrir de nouveaux toxicophores intéressants qui seraient manqués par les skypatterns.

6.6 Conclusion

Nous avons introduit la notion de skypattern souple et proposé une approche générique et efficace pour extraire les skypatterns ainsi que les skypatterns souples. De plus, le côté déclaratif du cadre de la PPC nous permet facilement de gérer les contraintes offrant plusieurs sortes de souplesse et conduit à un cadre unifié de manipulation de la souplesse dans le problème d'extraction des skypatterns. Enfin, la pertinence et l'efficacité de notre approche ont été mises en évidence par des expérimentations sur des jeux de données de l'*UCI* et une étude de cas en chemoinformatique pour la découverte de toxicophores.

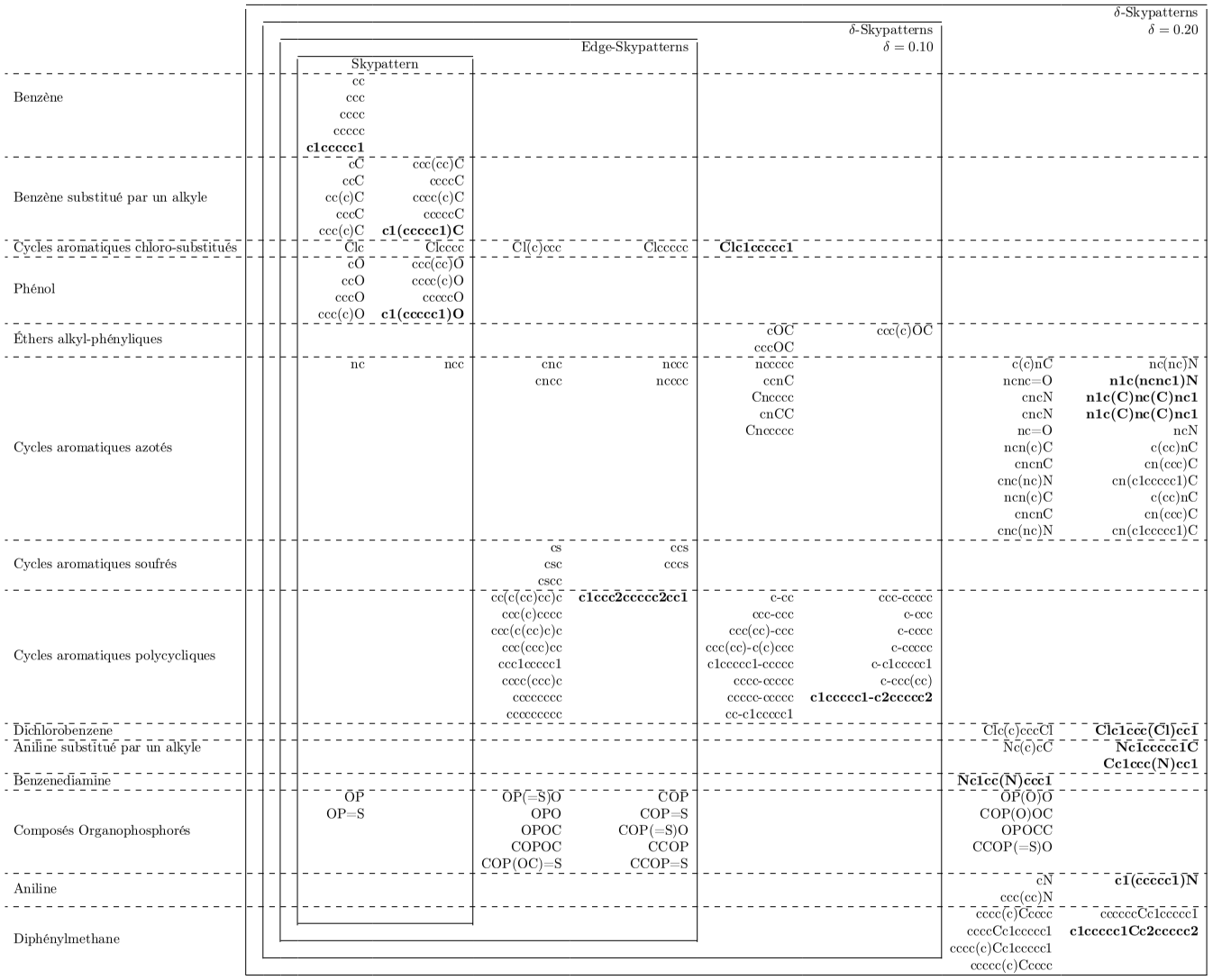


TABLE 6.9 – Classification des fragments toxicophores extraits pour M_4 selon le type de skypattern.

"Bien que ce soit de la folie, pourtant il y a une méthode en elle."

William Shakespeare, Hamlet

Chapitre 7

Cube de skypatterns

Sommaire

Dans la pratique, un utilisateur ne connaît pas le rôle exact de chaque mesure, et il lui est difficile de prévoir à l'avance l'ensemble de mesures le plus approprié. Idéalement, un utilisateur aimerait pouvoir examiner toutes les mesures potentiellement utiles, et, en supprimant ou en ajoutant une mesure, pouvoir évaluer son impact afin de converger vers les ensembles de skypatterns jugés les plus pertinents.

De plus, différents sous-ensembles de mesures peuvent avoir le même ensemble de skypatterns. Une telle classe d'équivalence sur les sous-ensembles de mesures peut permettre d'identifier les mesures peu pertinentes, voir inintéressantes pour le jeu de données traité (par exemple, les mesures qui peuvent être ajoutées/retirées d'un ensemble de mesures sans en changer l'ensemble de skypatterns).

Sur le même modèle que les cubes de skylines dans le bases de données (cf. la section 4.4), un utilisateur aimerait pouvoir disposer de cubes de skypatterns pour pouvoir explorer et affiner le rôle de chaque mesure et améliorer la qualité des motifs extraits.

Soit M un ensemble de mesures, son cube de skypatterns est constitué des $2^{|M|}-1$ ensembles de skypatterns associés aux sous-ensembles non-vides de M. Tous ces ensembles de skypatterns doivent être pré-calculés afin de pouvoir répondre rapidement aux différentes requêtes d'un utilisateur naviguant sur le cube. Une approche simple et naïve pour construire le cube serait de calculer séparément chacun de ces $2^{|M|}-1$ ensembles de skypatterns [36], mais le coût d'une telle approche serait prohibitif.

Ce chapitre s'intéresse à la construction du cube de skypatterns pour un ensemble de mesures M. La contribution de ce chapitre est double :

1. Nous proposons, tout d'abord, une méthode bottom-up fondée sur deux règles de dérivation de skypatterns permettant de déduire une grande majorité des skypatterns d'un nœud père à partir des skypatterns de ses nœuds fils [37]. Les skypatterns manquants d'un nœud père (i.e. ceux non-dérivables à partir de ses nœuds fils) sont alors calculés à la volée à l'aide des CSP dynamiques.
2. Puis, nous proposons une seconde méthode fondée sur la relaxation des skypatterns et utilisant les edge-skypatterns. Cette méthode repose sur le fait que tout ensemble de skypatterns d'un nœud est inclus dans l'ensemble des edge-skypatterns de M. Le problème de construction du cube de skypatterns peut alors être facilement ramené au calcul d'un cube de skylines en $|M|$ dimensions. Ce calcul est effectué par un extracteur existant de cubes de skylines.

A notre connaissance, ce sont les deux seules méthodes capables de construire efficacement un cube de skypatterns.

Ce chapitre est organisé comme suit. La section 7.1 introduit, pour un ensemble de mesures M, les notions de skypattern incomparable et de skypattern indistinct. Ces deux notions sont fondamentales pour l'élaboration de notre méthode bottom-up. En effet, chacune possède sa propre règle de dérivation des skypatterns d'un nœud père à partir des skypatterns de ses nœuds fils.

La section 7.2 présente notre méthode bottom-up en décrivant successivement nos règles de dérivation ainsi que le calcul à la volée des skypatterns non-dérivables à l'aide des CSP dynamiques.

La section 7.3 présente notre méthode fondée sur la relaxation en montrant que tout ensemble de skypatterns d'un nœud est inclus dans l'ensemble des edge-skypatterns de M. Le problème de construction du cube de skypatterns peut alors être facilement ramené au calcul d'une cube de skylines en $|M|$ dimensions. Ce calcul est effectué par un extracteur existant de cubes de skylines.

La section 7.4 et la section 7.5 présentent les expérimentations que nous avons menées sur le jeu de donnés réel Mutagénicité et sur différents jeux de données de l'*UCI*.

Enfin, dans ce chapitre, on dénotera désormais l'ensemble des skypatterns pour un ensemble de mesures M par : $Skypattern(M)$. Par rapport au chapitre précédent, on a : $Skypattern(M) = Sky(M, \mathcal{L}_{\mathcal{I}})$.

De même, on notera l'ensemble de skylines pour un ensemble de dimensions D par : $Skyline(D)$ (cf. le chapitre 4).

7.1 Définitions

Dans cette section, nous définissons, tout d'abord, la notion de skypatterns incomparables et de skypatterns indistincts (selon un ensemble de mesures M) et nous montrons qu'ils forment une partition de tout ensemble de skypatterns. Puis, nous définissons la notion de cube de skypatterns associé à un ensemble de mesures M.

36. Par exemple, en utilisant AETHERIS (cf. la section 6.1.2) ou bien CP+SKY (cf. la section 6.3).
37. Si k mesures sont associées à un nœud père, alors ses nœuds fils sont les nœuds définis par les $k-1$ sous-ensembles possibles de $(k-1)$ mesures.

7.1.1 Skypatterns incomparables et skypatterns indistincts

Soit M un ensemble de mesures, la notion de Pareto dominance a été introduite par la définition 6.1 (cf. page 86) : un motif $\mathtt{x}$ domine un autre motif $\mathtt{y}$ par rapport à M (dénoté par $\mathtt{x} \succ_M \mathtt{y}$) si et seulement si $\forall m \in M, m(\mathtt{x}) \geq m(\mathtt{y})$ et $\exists m' \in M, m'(\mathtt{x}) > m'(\mathtt{y})$. De même, la notion de skypattern a été introduite par la définition 6.2 (cf. page 87) : un skypattern est un motif non dominé selon M.

Transaction	Classe	Items					
t_1	j_1	A				E	F
t_2	j_1		B	C	D	E	
t_3	j_1		B	C	D	E	F
t_4	j_2	A	B	C	D	E	
t_5	j_2		B	C	D		
t_6	j_2		B			E	F
t_7	j_2	A	B	C	D	E	F

Item	Prix
A	30
B	40
C	10
D	40
E	70
F	55

TABLE 7.1 – Jeu de données transactionnel $\boldsymbol{r}$.

Exemple 7.1.
Pour le jeu de données du tableau 7.1, avec $M = \{\mathtt{freq}, \mathtt{aire}\}$, BCD domine BC (noté $BCD \succ_M BC$) car $\mathtt{freq}(BCD) = \mathtt{freq}(BC) = 5$ et $\mathtt{aire}(BCD) > \mathtt{aire}(BC)$.
La figure 7.1 fournit une représentation graphique de $Skypattern(M) = \{BCDE, BCD, B, E\}$.

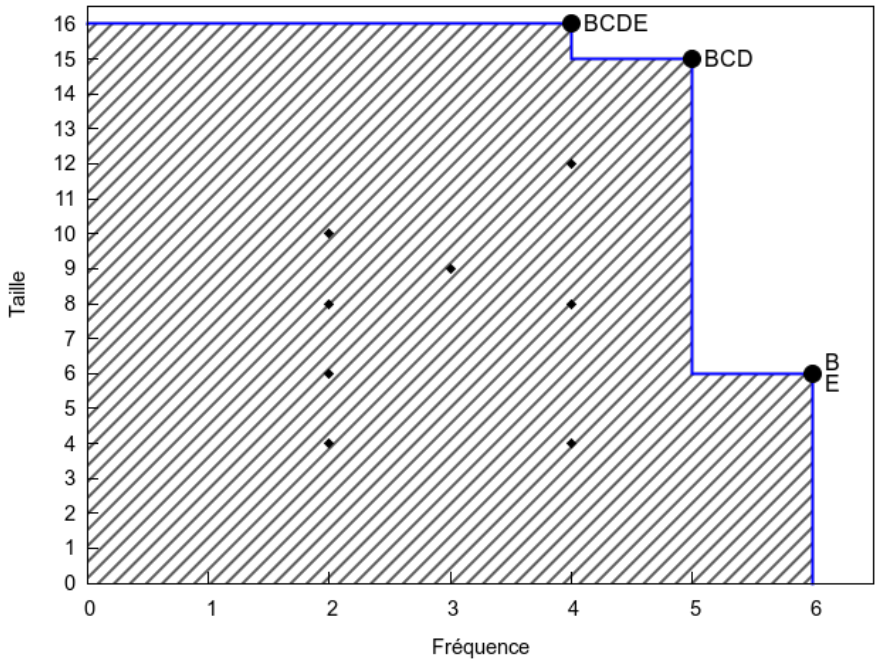


FIGURE 7.1 – Skypatterns pour $M = \{\mathtt{freq}, \mathtt{aire}\}$.

Définition 7.1. Motifs indistincts
Étant donné un ensemble de mesures M, deux motifs $\mathtt{x}$ et $\mathtt{y}$ sont *indistincts* par rapport à M (dénoté $\mathtt{x} =_M \mathtt{y}$) si et seulement si $\forall m \in M, m(\mathtt{x}) = m(\mathtt{y})$.

Définition 7.2. Motifs incomparables
Étant donné un ensemble de mesures M, deux motifs $\mathtt{x}$ et $\mathtt{y}$ sont *incomparables* par rapport à M (dénoté $\mathtt{x} \prec\succ_M \mathtt{y}$) si et seulement si ($\mathtt{x} \not\succ_M \mathtt{y}$) et ($\mathtt{y} \not\succ_M \mathtt{x}$) et ($\mathtt{x} \neq_M \mathtt{y}$).

Ces deux définitions peuvent être étendues aux skypatterns.

Définition 7.3. Skypattern incomparable
Étant donné un ensemble de mesures M, un motif $\mathtt{x} \in Skypattern(M)$ est incomparable par rapport à M si et seulement si $\forall \mathtt{y} \in Skypattern(M)$ tel que $\mathtt{y} \neq \mathtt{x}$, $\mathtt{x} \prec\succ_M \mathtt{y}$.

Définition 7.4. Skypattern indistinct
Étant donné un ensemble de mesures M, un motif $\mathtt{x} \in Skypattern(M)$ est indistinct par rapport à M si et seulement si $\exists \mathtt{y} \in Skypattern(M)$ tel que $\mathtt{y} \neq \mathtt{x}$, $\mathtt{x} =_M \mathtt{y}$.

Les skypatterns incomparables et les skypatterns indistincts par rapport à M constituent une partition de $Skypattern(M)$. De plus, $=_M$ est une relation d'équivalence (i.e. la relation est réflexive, symétrique et transitive). Les skypatterns indistincts peuvent donc être réunis dans un même groupe.

Définition 7.5. Groupe de skypatterns indistincts (GSI)
$S \subseteq Skypattern(M)$ est un *groupe de skypatterns indistincts* par rapport à M, si et seulement si $|S| \geq 2$ et $\forall \mathtt{x}, \mathtt{y} \in S, (\mathtt{x} =_M \mathtt{y})$ et $\forall \mathtt{x} \in S, \forall \mathtt{y} \in Skypattern(M) \setminus S, (\mathtt{x} \prec\succ_M \mathtt{y})$.

Exemple 7.2.
Pour $M = \{\mathtt{freq}, \mathtt{aire}\}$, on a $Skypattern(M) = \{BCDE, BCD, B, E\}$ (cf. la figure 7.1). Les skypatterns $BCDE$ et BCD sont incomparables, alors que les skypatterns B et E sont indistincts et appartiennent au même GSI.

7.1.2 Cube de skypatterns

Soit M un ensemble de mesures, le cube de skypatterns sur M se compose des ensembles de skypatterns de M_u, pour tous les sous-ensembles non-vides M_u de M.

Définition 7.6. Cube de skypatterns
Étant donné un ensemble de mesures M,

$$Skypattern\text{-}Cube(M) = \{(M_u, Skypattern(M_u)) \mid \ M_u \subseteq M, M_u \neq \emptyset\}$$

Exemple 7.3.
Considérons le jeu de données du tableau 7.1, la figure 7.2 représente le treillis associé à M et le tableau 7.2 associe, à chaque sous-ensemble non-vide de M, son ensemble de skypatterns.

Différents sous-ensembles de mesures peuvent conduire à un même ensemble de skypatterns. Cette observation peut être utilisée pour fournir une représentation concise du cube sans perte d'information.

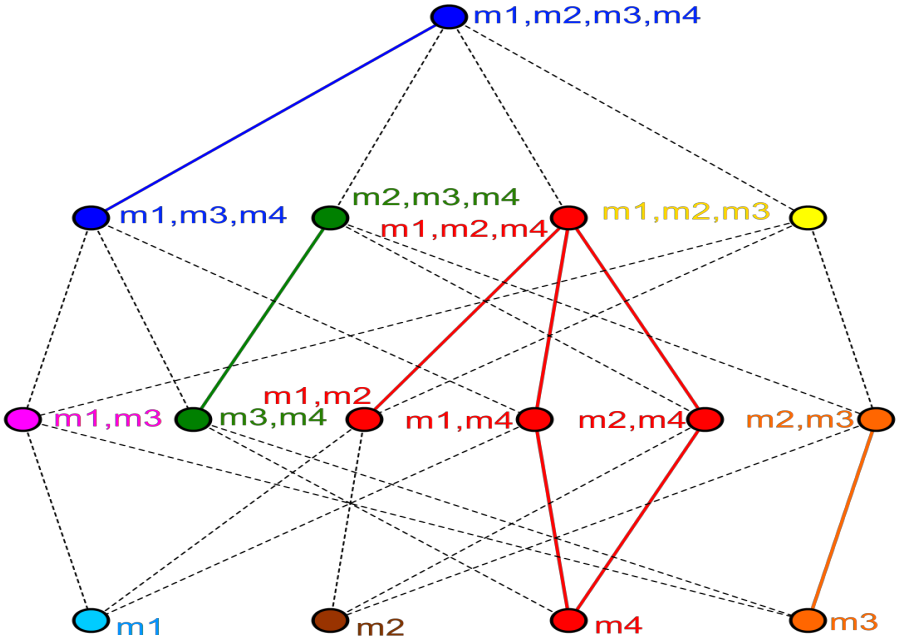


FIGURE 7.2 – Treillis de mesures associé à $M = \{m_1 : \texttt{freq}, m_2 : \texttt{émergence}_\text{j}, m_3 : \texttt{aire}, m_4 : \texttt{mean}\}$.

Subset of M	**Skypattern set**
$\{m_1, m_2, m_3, m_4\}$	{BCD, BCDE, BDE, BE, BEF, E, EF}
$\{m_1, m_2, m_3\}$	{BCD, BCDE, E}
$\{m_1, m_2, m_4\}$	{E}
$\{m_1, m_3, m_4\}$	{BCD, BCDE, BDE, BE, BEF, E, EF}
$\{m_2, m_3, m_4\}$	{BCDE, BDE, BE, BEF, E, EF}
$\{m_1, m_2\}$	{E}
$\{m_1, m_3\}$	{BCD, BCDE, B, E}
$\{m_1, m_4\}$	{E}
$\{m_2, m_3\}$	{BCDE}
$\{m_2, m_4\}$	{E}
$\{m_3, m_4\}$	{BCDE, BDE, BE, BEF, E, EF}
$\{m_1\}$	{B, E}
$\{m_2\}$	{AEF, AF, BCDE, BCDEF, BCDF, BDE, BDEF, BDF, E, EF, F}
$\{m_3\}$	{BCDE}
$\{m_4\}$	{E}

TABLE 7.2 – Cube de skypatterns pour $M = \{m_1 : \texttt{freq}, m_2 : \texttt{émergence}_\text{j}, m_3 : \texttt{aire}, m_4 : \texttt{mean}\}$.

Nous définissons une relation d'équivalence sur les sous-ensembles de mesures ayant le même ensemble de skypatterns :

Définition 7.7. ÉQUIVALENCE ENTRE ENSEMBLES DE MESURES
Étant donné un ensemble de mesures M, et soient $M_u \subseteq M$ et $M_v \subseteq M$. M_u et M_v sont équivalents si et seulement si $Skypattern(M_u) = Skypattern(M_v)$.

Exemple 7.4.
Les classes d'équivalence sur notre exemple sont illustrées par la figure 7.2. Il y a huit classes d'équivalence :

une est de cardinalité 5 (en rouge), trois sont de cardinalité 2 (en bleu foncé, vert et orange), et les quatre restantes sont de cardinalité 1.

7.2 Approche bottom-up

Cette section présente notre approche bottom-up pour construire un cube de skypatterns et constitue la première contribution de ce chapitre.

Tout d'abord, nous proposons deux règles de dérivation (l'une pour les skypatterns incomparables, l'autre pour les skypatterns indistincts) permettant de déduire automatiquement une grande partie des skypatterns d'un nœud père à partir des ensembles de skypatterns de ses nœuds fils, ceci sans effectuer aucun test de dominance.

Puis, les skypatterns manquants d'un nœud père (i.e. ceux non-dérivables à partir de ses nœuds fils) sont alors calculés à la volée à l'aide des CSP dynamiques. Nous constaterons expérimentalement que le nombre de skypatterns non-dérivables demeure faible en pratique.

Enfin, nous montrons comment notre approche bottom-up permet de déterminer les classes d'équivalence sans aucun effort supplémentaire. Ce résultat a l'avantage de fournir une représentation plus concise du cube, en mettant en évidence les mesures ayant le même ensemble de skypatterns.

A notre connaissance, cette approche bottom-up est la première méthode non-naïve et efficace pour construire un cube de skypatterns.

7.2.1 Règles de dérivation

Cette section présente nos deux règles de dérivation permettant la construction ascendante du cube de skypatterns. Elles permettent de déduire, sans effectuer le moindre test de dominance, une grande partie (cf. la section 7.4.2-β) des skypatterns d'un nœud père à l'aide des skypatterns de ses nœuds fils.

7.2.1a A partir des incomparables d'un nœud fils

La première règle de dérivation porte sur la notion de sypattern incomparable : tous les skypatterns incomparables d'un nœud fils sont aussi des skypatterns incomparables pour ses nœuds pères (cf. le théorème 3).

Théorème 3 *Règle d'incomparabilité.* Soit $M_u \subseteq M$, si $\mathtt{x}$ est un skypattern incomparable par rapport à M_u alors $\forall m \in M \setminus M_u$, $\mathtt{x}$ est un skypattern (incomparable) par rapport à $M_u \cup \{m\}$.

Par contradiction. Supposons que $\mathtt{x}$ est un skypattern incomparable par rapport à M_u, et $\exists m \in M \setminus M_u$ tel que $\mathtt{x} \notin Skypattern(M_u \cup \{m\})$.

Alors, $\exists \mathtt{y} \neq \mathtt{x} \in \mathcal{L}_{\mathcal{I}}$, ($\mathtt{y} \succ_{M_u \cup \{m\}} \mathtt{x}$) i.e. (1) $\forall m' \in M_u \cup \{m\}, m'(\mathtt{y}) \geq m'(\mathtt{x})$ et (2) $\exists m'' \in M_u \cup \{m\}$, $m''(\mathtt{y}) > m''(\mathtt{x})$. Pour (2), il y a deux cas :

1. ($m'' = m$). Comme $\mathtt{x} \in Skypattern(M_u)$, $\forall m' \in M_u, m'(\mathtt{x}) \geq m'(\mathtt{y})$. De (1), on déduit que : $\forall m' \in M_u, m'(\mathtt{x}) = m'(\mathtt{y})$. Donc $\mathtt{x}$ est indistinct par rapport à M_u. Ce qui contredit que $\mathtt{x}$ soit incomparable par rapport à M_u.
2. ($m'' \in M_u$). De (1), on a que : $\forall m' \in M_u, m'(\mathtt{y}) \geq m'(\mathtt{x})$. Comme, $m''(\mathtt{y}) > m''(\mathtt{x})$, on en déduit que $\mathtt{y} \succ_{M_u} \mathtt{x}$. Ce qui contredit que $\mathtt{x}$ soit un skypattern par rapport à M_u.

Enfin, $\mathtt{x}$ étant incomparable par rapport à M_u, $\mathtt{x}$ sera donc incomparable par rapport à $M_u \cup \{m\}$. □

Exemple 7.5.
Soit $M_u = \{m_1, m_3\}$, $BCDE$ et BCD sont incomparables par rapport à M_u. Le théorème 3 permet de déduire que $BCDE$ et BCD appartiennent à $Skypattern(M_u \cup \{m_2\})$ et $Skypattern(M_u \cup \{m_4\})$.

7.2.1b A partir des indistincts d'un nœud fils

Mais, si pour les skypatterns incomparables la règle de dérivation est simple, il n'en est pas de même pour la notion de skypattern indistinct. En effet, un skypattern indistinct pour un nœud fils :
- peut devenir un skypattern incomparable pour un nœud père,
- peut demeurer un skypattern indistinct pour un nœud père,
- ne plus être un skypattern pour un nœud père.

Le théorème 4 et le théorème 5 caractérisent la nature, pour un nœud père, des skypatterns qui sont indistincts pour l'un de ses nœuds fils.

Théorème 4 *Règle pour les GSI.* Soient $M_u \subseteq M$ et S un GSI par rapport à M_u. $\forall m \in M \setminus M_u$, chaque skypattern $\mathtt{x} \in S$ tel que $m(\mathtt{x}) = \max_{\mathtt{x}' \in S} \{m(\mathtt{x}')\}$ est un skypattern par rapport à $M_u \cup \{m\}$.

Par contradiction. Supposons que il existe un GSI S par rapport à M_u tel que $\exists m \in M \setminus M_u$ et $\exists \mathtt{x} \in S$ tel que $m(\mathtt{x}) = \max_{\mathtt{x}' \in S} \{m(\mathtt{x}')\}$ et $\mathtt{x} \notin Skypattern(M_u \cup \{m\})$. Alors, $\exists \mathtt{y} \neq \mathtt{x} \in \mathcal{L}_{\mathcal{I}}$, $(\mathtt{y} \succ_{M_u \cup \{m\}} \mathtt{x})$ i.e. (1) $\forall m' \in M_u \cup \{m\}, m'(\mathtt{y}) \geq m'(\mathtt{x})$ et (2) $\exists m'' \in M_u \cup \{m\}, m''(\mathtt{y}) > m''(\mathtt{x})$. Pour (2), il y a 2 cas :

1. $(m'' = m)$. Comme $\mathtt{x} \in Skypattern(M_u)$, $\forall m' \in M_u, m'(\mathtt{x}) \geq m'(\mathtt{y})$. De (1), on déduit que : $\forall m' \in M_u, m'(\mathtt{x}) = m'(\mathtt{y})$, i.e. $\mathtt{y} \in S$. Donc, $m(\mathtt{y}) \leq m(\mathtt{x})$ (comme $m(\mathtt{x}) = \max_{\mathtt{x}' \in S} \{m(\mathtt{x}')\}$. Ce qui contredit (2).
2. $(m'' \in M_u)$. De (1), on a que $\forall m' \in M_u, m'(\mathtt{y}) \geq m'(\mathtt{x})$. Comme, $m''(\mathtt{y}) > m''(\mathtt{x})$, on déduit que : $\mathtt{y} \succ_{M_u} \mathtt{x}$. Ce qui contredit que $\mathtt{x}$ soit un skypattern par rapport à M_u.

□

Exemple 7.6.
$S = \{B, E\}$ est un GSI par rapport à $\{m_1\}$. Le théorème 4 permet de déduire que :
- $E \in Skypattern(\{m_1, m_2\})$ car $m_2(E) = \max_{\mathtt{x}' \in S} \{m_2(\mathtt{x}')\}$
- $E \in Skypattern(\{m_1, m_4\})$ car $m_4(E) = \max_{\mathtt{x}' \in S} \{m_4(\mathtt{x}')\}$
- $B, E \in Skypattern(\{m_1, m_3\})$ car $m_3(B) = m_3(E) = \max_{\mathtt{x}' \in S} \{m_3(\mathtt{x}')\}$

Théorème 5 *Statut des indistincts.* Soient $M_u \subseteq M$, $m \in M \setminus M_u$ et S un GSI par rapport à M_u, on définit $S' = \{\mathtt{x} \in S \mid \ m(\mathtt{x}) = \max_{\mathtt{x}' \in S} \{m(\mathtt{x}')\}\}$.
Si S' est un singleton alors l'unique skypattern est incomparable par rapport à $M_u \cup \{m\}$ sinon S' est un GSI par rapport à $M_u \cup \{m\}$. Enfin, tous les $\mathtt{x} \in S \setminus S'$ ne sont pas des skypatterns pour $M_u \cup \{m\}$.

Démonstration. Si le maximum est unique, alors ce motif est un skypattern incomparable pour $M_u \cup \{m\}$. Sinon, tous ces motifs sont des skypatterns indistincts par rapport à $M_u \cup \{m\}$. En effet, (1) ils ont tous la même valeur pour la mesure m, (2) ils ont aussi tous la même valeur pour toute mesure $m' \in M_u$ car S est un GSI pour M_u.
Soit $\mathtt{x} \in S \setminus S'$, le motif $\mathtt{x}$ ne peut être un skypattern par rapport à $M_u \cup \{m\}$; en effet, car chaque $\mathtt{x}' \in S'$ domine $\mathtt{x}$ car $m(\mathtt{x}') > m(\mathtt{x})$ et $\forall m' \in M_u, m'(\mathtt{x}') = m'(\mathtt{x})$. □

7.2.1c Qualité des règles de dérivation

Les deux règles de dérivation que nous proposons possèdent deux propriétés importantes qui illustrent leur qualité :

1. Elles permettent de dériver une grande partie des skypatterns d'un nœud père à l'aide des skypatterns de ses nœuds fils (cf. la section 7.4.2-β).
2. Si un skypattern d'un nœud père est non-dérivable par ces deux règles, alors il n'est un skypattern pour aucun de ses nœuds fils.

Soient $M_u \subseteq M$ et $m \in M \setminus M_u$, on définit :

— $Incomparable(M_u)$ qui est l'ensemble de skypatterns incomparables par rapport à M_u

$$Incomparable(M_u) = \{\mathtt{x} \in Skypattern(M_u) \mid \mathtt{x} \text{ est incomparable par rapport à } M_u\}$$

— $Indistinct(M_u, m)$ qui est l'ensemble de skypatterns indistincts maximaux par rapport à m :

$$Indistinct(M_u, m) = \bigcup_{GSI\ S \subseteq Skypattern(M_u)} \{\mathtt{x} \in S \mid m(\mathtt{x}) = \max_{\mathtt{x}' \in S} \{m(\mathtt{x}')\}\}$$

On définit alors $Derived(M_u)$ qui est l'ensemble de skypatterns par rapport à M_u qui peuvent être dérivés à partir des skypatterns de ses nœuds fils :

$$Derived(M_u) = \bigcup_{m \in M_u} (Incomparable(M_u \setminus \{m\}) \cup Indistinct(M_u \setminus \{m\}, m))$$

Tout d'abord, il est immédiat que : $Derived(M_u) \subseteq Skypattern(M_u)$ (cf. les théorèmes 3 et 4). Les expérimentations que nous avons menées montrent qu'une grande proportion de skypatterns d'un nœud père sont obtenus de cette façon (cf. la section 7.4.2-δ).

De plus, si un skypattern $\mathtt{x}$ d'un nœud père est également un skypattern pour au moins un de ses nœuds fils, alors $\mathtt{x}$ sera nécessairement dérivé par l'une des deux règles. La preuve de cette propriété est immédiate car, pour chaque nœud, les skypatterns incomparables et les skypatterns indistincts constituent une partition de l'ensemble des skypatterns. On a alors :

$$Derived(M_u) = \left(\bigcup_{m \in M_u} Skypattern(M_u \setminus \{m\}) \right) \cap Skypattern(M_u)$$

Cette propriété illustre la qualité de nos deux règles de dérivation : si un skypattern d'un nœud père est non-dérivable par ces deux règles, alors il n'est un skypattern pour aucun de ses nœuds fils.

Ce qui montre que nos deux règles de dérivation dérivent "tout ce qui peut être dérivé".

7.2.2 Skypatterns non-dérivables

Les skypatterns d'un nœud père sont obtenus en deux étapes : (1) tous les skypatterns qui peuvent être dérivés de ses nœuds fils sont calculés ; (2) les skypatterns manquants (i.e. les skypatterns non-dérivables) sont calculés à la volée en utilisant les CSP dynamiques. Dans cette section, nous proposons une condition suffisante permettant de déterminer la non-existence de skypatterns non-dérivables pour un nœud donné. L'intérêt de cette CS est d'éviter de lancer le calcul des non-dérivables alors que l'on peut déterminer a priori qu'il n'en existe aucun.

Tout d'abord, tous les skypatterns d'un nœud père ne sont pas forcément dérivables. C'est le cas lorsqu'un skypattern d'un nœud père n'est un skypattern pour aucun de ses nœuds fils.

Exemple 7.7.
Soit $M = \{m_1 : \texttt{freq}, m_3 : \texttt{aire}\}$, considérons la figure 7.3. Comme $BCDE$ est incomparable par rapport à $\{m_3\}$, $BCDE \in Skypattern(\{m_1, m_3\})$. Comme B et E constituent un GSI par rapport à $\{m_1\}$ et $m_3(B) = m_3(E)$, alors $B, E \in Skypattern(\{m_1, m_3\})$. Mais, les règles de dérivation ne peuvent pas déduire que $BCD \in Skypattern(\{m_1, m_3\})$.
On a $Derived(\{m_1, m_3\}) = \{B, E, BCDE\}$ alors que $Skypattern(\{m_1, m_3\}) = \{B, E, BCD, BCDE\}$.

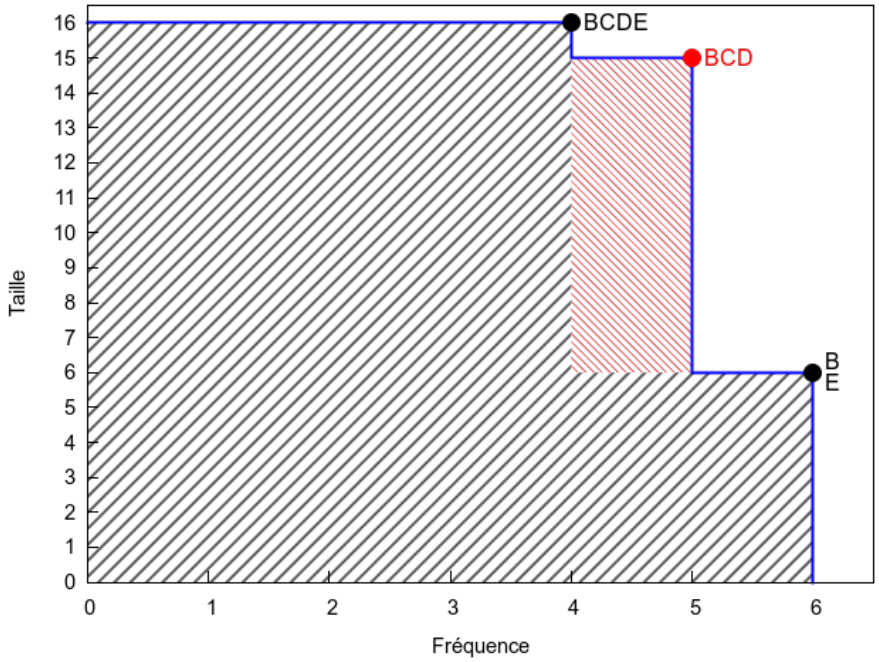


FIGURE 7.3 – BCD est un skypattern non-dérivable pour $M = \{m_1 : \texttt{freq}, m_3 : \texttt{aire}\}$.

Soit M un ensemble de mesures et $M_u \subseteq M$, le théorème 6 procure une condition suffisante assurant que $Derived(Mu) = Skypattern(M_u)$, i.e. qu'il n'existe aucun skypattern non-dérivable pour M_u. Les expérimentations montrent que cette condition est efficace dans la pratique (cf. la section 7.4.2-γ).

Théorème 6 *CS de non-existence de skypatterns non dérivables.* Soient M un ensemble de mesures et $M_u \subseteq M$.
Si $\exists m \in M_u, \min\limits_{\texttt{x}' \in Derived(M_u)} \{m(\texttt{x}')\} = \max\limits_{\texttt{x}'' \in Derived(M_u)} \{m(\texttt{x}'')\}$ alors $Skypattern(M_u) = Derived(Mu)$.

Par contradiction. Soit $m \in M_u$ une mesure tel que $\min\limits_{\texttt{x}' \in Derived(M_u)} \{m(\texttt{x}')\} = \max\limits_{\texttt{x}'' \in Derived(M_u)} \{m(\texttt{x}'')\}$
Supposons que $\exists \texttt{x} \in Skypattern(M_u) \setminus Derived(M_u)$, ainsi

$$m(\texttt{x}) = \min_{\texttt{x}' \in Derived(M_u)} \{m(\texttt{x}')\} = \max_{\texttt{x}'' \in Derived(M_u)} \{m(\texttt{x}'')\}$$

Dorénavant $\texttt{x} \in Skypattern(\{m\})$ (i.e. $m(\texttt{x}) = \max\limits_{\texttt{x}''' \in \mathcal{L}_\mathcal{I}} \{m(\texttt{x}''')\}$).
Sinon, $\exists \texttt{y} \in \mathcal{L}_\mathcal{I}, \texttt{y} \in Skypattern(\{m\})$ (i.e. $m(\texttt{y}) = \max\limits_{\texttt{x}''' \in \mathcal{L}_\mathcal{I}} \{m(\texttt{x}''')\}$) tel que $m(\texttt{y}) > \max\limits_{\texttt{x}'' \in Derived(M_u)} \{m(\texttt{x}'')\}$,
mais comme $\texttt{y} \in Skypattern(\{m\})$, on a que $\texttt{y} \in Derived(M_u)$, d'où $m(\texttt{y}) = \max\limits_{\texttt{x}'' \in Derived(M_u)} \{m(\texttt{x}'')\}$.

Par conséquent, $\mathtt{x} \in Skypattern(\{m\})$:

(i) soit $\mathtt{x}$ est incomparable dans $Skypattern(\{m\})$,

(ii) ou $\mathtt{x}$ est indistinct dans $Skypattern(\{m\})$ avec la valeur maximale de m.
A partir de (i) et (ii), $\mathtt{x} \in Derived(M_u)$ conduisant à une contradiction. □

7.2.3 Construction du cube et de sa représentation concise

Cette section décrit tout d'abord le pseuod-code de notre approche bottom-up pour construire un cube de skypatterns. Puis, elle montre comment les classes d'équivalence peuvent être déterminées, sans aucun effort supplémentaire, tout au long de la construction ascendante du cube afin d'en produire une représentation concise (cf. la section 7.1.2).

L'algorithme 5 donne le pseudo-code de notre approche bottom-up. Tout d'abord, pour chaque mesure $m \in M$, on calcule son ensemble de skypatterns. Puis l'approche bottom-up permet de construire le cube niveau par niveau. Les ensembles de skypatterns de chaque niveau i du treillis sont obtenus en appliquant les règles de dérivation et, si nécessaire, le calcul des skypatterns non-dérivables (fonction $Complete$), aux ensembles de skypatterns obtenus au niveau précédent $(i-1)$. L'algorithme s'arrête après avoir déterminé $Skypattern(M)$.

Algorithme 5 : Approche ascendante pour le calcul du cube de skypatterns.

Entrées : M : un ensemble de mesures, $\mathcal{T}$: un jeu de données.
Sorties : Le cube de skypatterns de $\mathcal{T}$ par rapport à M.

```
1 cube ← ∅;
2 pour chaque m ∈ M faire
3     cube ← cube ∪ {({m}, Skypattern({m}))};
4 pour i ← 2 à |M| faire
5     pour chaque M_u ⊂ M tel que |M_u| = i faire
6         Derived(M_u) ← ⋃_{m∈M_u} (Incomparable(M_u \ {m}) ∪ Indistinct(M_u \ {m}, m));
7         cube ← cube ∪ {(M_u, Complete(Derived(M_u)))};
8 retourner cube
```

Le théorème 7 indique si un nouveau nœud construit par l'ajout d'une mesure à un sous-ensemble de mesures M_u appartient ou non à la classe d'équivalence de M_u. Les classes d'équivalence peuvent alors être facilement déterminées lors de la construction ascendante du cube de skypatterns.

Théorème 7 *Caractérisation des classes d'équivalence.* Soient M un ensemble de mesures, $M_u \subseteq M$ et $m \in M \setminus M_u$, $Skypattern(M_u \cup \{m\}) = Skypattern(M_u)$ si et seulement si
(1) tous les skypatterns indistincts par rapport à M_u sont des skypatterns indistincts pour $M_u \cup \{m\}$ et (2) $Skypattern(M_u \cup \{m\}) = Derived(M_u \cup \{m\})$.

Double inclusion. Tous les incomparables par rapport à M_u sont incomparables par rapport à $M_u \cup \{m\}$ (cf. le théorème 3). Tous les indistincts par rapport à M_u sont indistinct par rapport à $M_u \cup \{m\}$ selon (1). Or, les incomparables par rapport à M_u et les indistincts par rapport à M_u forment une partition de $Skypattern(M_u)$. On a donc $Skypattern(M_u) \subset Skypattern(M_u \cup \{m\})$.
Selon (2), $Skypattern(M_u \cup \{m\}) = Derived(M_u \cup \{m\})$. Comme les skypatterns dérivés par rapport à $M_u \cup \{m\}$ ne peuvent venir que de $Skypattern(M_u)$, on a donc $Skypattern(M_u \cup \{m\}) \subset Skypattern(M_u)$.
□

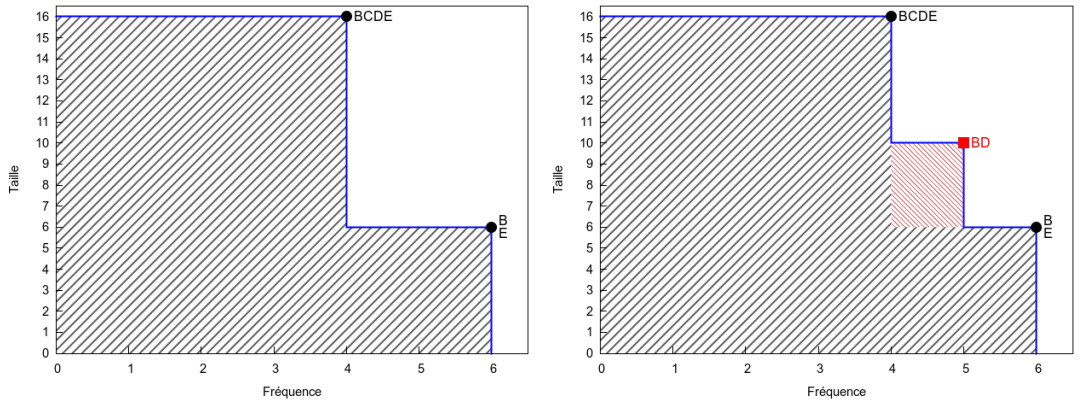
7.2.4 Extraction des non-dérivables à l'aide des CSP dynamiques

Cette section décrit comment les skypatterns non-dérivables peuvent être extraits à la volée en utilisant les CSP dynamiques. Notre approche s'inspire de l'extraction des skypatterns à l'aide des CSP dynamiques présentée à la section 6.3.1 (cf. page 92). Tout d'abord, on impose que le motif recherché $\mathtt{x}$ ne soit dominé par aucun des skypatterns déjà dérivés. Puis, chaque requête $\mathtt{q}_{i+1}(\mathtt{x})$ impose que le motif recherché $\mathtt{x}$ ne soit dominé par aucun des candidats déjà obtenus : $s_1, \ldots, s_i$.

Soient M un ensemble de mesures, $M_u \subseteq M$, M'_u l'ensemble de mesures "skylineables" de M_u (cf. le principe, provenant d'AETHERIS, des mesures skylineables est explicité à la section 6.1.2) et $Derived(M_u)$ le sous-ensemble de $Skypattern(M_u)$ obtenu en appliquant les deux règles de dérivation présentées à la section 7.2.1.

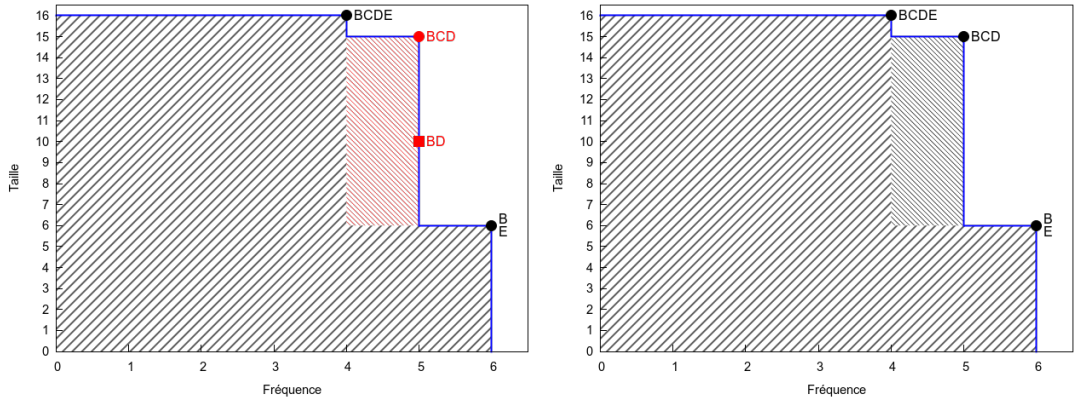
Considérons la séquence $P_1, \ldots, P_n$ de CSP où chaque $P_i = (\{\mathtt{x}\}, \mathcal{L}_\mathcal{I}, \mathtt{q}_i(\mathtt{x}))$ et :

— $\mathtt{q}_1(\mathtt{x}) = \mathtt{fermé}_{M'_u}(\mathtt{x}) \wedge \Upsilon_{M_u}(\mathtt{x})$
— $\mathtt{q}_{i+1}(\mathtt{x}) = \mathtt{q}_i(\mathtt{x}) \wedge (s_i \not\succ_{M_u} \mathtt{x})$ où s_i est la première solution à la requête $\mathtt{q}_i(\mathtt{x})$



(a) Départ avec les 3 skypatterns dérivés.

(b) Obtention du 1er candidat : BD.



(c) Obtention du 2nd candidat : BCD.

(d) Seul le 2nd candidat est un skypattern.

FIGURE 7.4 – Calcul des non-dérivables à l'aide des CSP dynamiques.

où $\Upsilon_{M_u}(\mathtt{x})$ stipules que $\mathtt{x}$ ne peut pas être dominé par rapport à M_u par aucun skypattern derivé :

$$\Upsilon_{M_u}(\mathtt{x}) = \bigwedge_{\mathtt{x}' \in Derived(M_u)} (\mathtt{x}' \not\succ_{M_u} \mathtt{x})$$

Exemple 7.8.
Considérons l'exemple 7.7. Pour $M_u = \{m_1 : \mathtt{freq}, m_3 : \mathtt{aire}\}$, $Derived(M_u) = \{B, E, BCDE\}$. Le motif $BCDE$ est incomparable alors que les motifs B et E sont indistincts.

Donc $\Upsilon_{M_u}(\mathtt{x}) = (B \not\succ_{M_u} \mathtt{x}) \wedge (E \not\succ_{M_u} \mathtt{x}) \wedge (BCDE \not\succ_{M_u} \mathtt{x})$. Pour M_u, il existe un seul skypattern non-dérivable : $\mathtt{x} = BCD$.

La figure 7.4 illustre ce calcul. La requête initiale $\mathtt{q}_1(\mathtt{x})$ impose que $\mathtt{x}$ ne soit dominé par aucun des motifs dérivés. On a donc : $\mathtt{q}_1(\mathtt{x}) \equiv (B \not\succ_{M_u} \mathtt{x}) \wedge (E \not\succ_{M_u} \mathtt{x}) \wedge (BCDE \not\succ_{M_u} \mathtt{x})$.

La première solution de $\mathtt{q}_1(\mathtt{x})$ est le motif BD. La seconde requête $\mathtt{q}_2(\mathtt{x})$ impose que $\mathtt{x}$ ne soit pas non plus dominé par BD. On a donc : $\mathtt{q}_2(\mathtt{x}) \equiv \mathtt{q}_1(\mathtt{x}) \wedge (BD \not\succ_{M_u} \mathtt{x})$.

La première solution de $\mathtt{q}_2(\mathtt{x})$ est le motif BCD. La requête suivante $\mathtt{q}_3(\mathtt{x})$ impose que $\mathtt{x}$ ne soit pas non plus dominé par BCD. On a donc : $\mathtt{q}_3(\mathtt{x}) \equiv \mathtt{q}_2(\mathtt{x}) \wedge (BCD \not\succ_{M_u} \mathtt{x})$. Le processus s'arrête car cette requête ne possède pas de solution.

Mais, comme pour l'extraction des skypatterns (cf. la section 6.3.1 à la page 92) et l'extraction des skypatterns souples (cf. la section 6.3.3 à la page 94), tous les candidats ne sont pas obligatoirement des skypatterns. D'où l'obligation d'effectuer un filtrage de l'ensemble des candidats. Ici, le motif BD n'est pas un skypattern pour M_u car il est dominé par BCD [38]. On retrouve bien le résultat de la figure 7.3.

7.3 Approche par relaxation

Cette section présente une seconde approche pour construire le cube de skypatterns associé à un ensemble de mesures M. Cette approche est fondée sur la relaxation dans le processus d'extraction de motifs. Elle constitue la seconde contribution de ce chapitre.

Notre approche est fondée sur le fait que chaque nœud du cube peut être approximé par un ensemble unique, *Edge-Skypattern*(M), qui est l'ensemble des edge-skypatterns par rapport à M. En effet, pour tout $M_u \subseteq M$, on a *Skypattern*$(M_u) \subseteq$ *Edge-Skypattern*(M) (cf. le théorème 9).

Le problème de calcul du cube de skypatterns peut alors être transformé en un problème (équivalent) de calcul d'un cube de skylines en $|M|$ dimensions. Ce problème est résolu par un extracteur de cubes de skylines (cf. la section 4.4 à la page 55) et le cube de skypatterns recherché en est alors déduit.

A notre connaissance, cette approche fondée sur la relaxation est la seconde méthode non-naïve et efficace pour construire un cube de skypatterns.

La sous-section 7.3.1 fournit "le pourquoi et le comment" de notre approche fondée sur la relaxation. La sous-section 7.3.2 décrit comment *Edge-Skypattern*(M) peut être efficacement extrait en utilisant soit une méthode fondée sur les CSP dynamiques, soit une version étendue de AETHERIS. Enfin, la sous-section 7.3.3 montre comment le calcul du cube de skypatterns (selon M) peut ensuite être converti en un calcul d'un cube de skylines (en $|M|$ dimensions).

38. $\mathtt{freq}(BD) = \mathtt{freq}(BCD) = 5$ et $\mathtt{aire}(BD) < \mathtt{aire}(BCD)$.

Pattern	m_1	m_2	m_3	m_4
AEF	2	1.33	6	50
AF	2	1.33	4	42
BCD	5	0.88	15	25
BCDE	4	1.33	16	40
BCDEF	2	1.33	10	40
BCDF	2	1.33	8	32
B	6	0.66	6	40
BD	5	0.88	10	40
BDE	4	1.33	12	55
BDEF	2	1.33	8	55
BDF	2	1.33	6	47
BE	5	0.88	10	55
BEF	3	0.66	9	55
E	6	1.33	6	70
EF	4	1.33	8	62
F	4	1.33	4	55

TABLE 7.3 – *Edge-Skypattern*(M) pour $M = \{m_1 : \texttt{freq}, m_2 : \texttt{émergence}_\text{j}, m_3 : \texttt{aire}, m_4 : \texttt{mean}\}$.

7.3.1 Approximer chaque nœud par *Edge-Skypattern*(M)

7.3.1a Résultat principal

Soit M un ensemble de mesures, contrairement à l'opérateur *Sky*, l'opérateur *Edge-Sky* est monotone (cf. le théorème 8). En conséquence, chaque nœud du cube est inclus dans *Edge-Skypattern*(M) (cf. le théorème 9). C'est le résultat principal sur lequel se fonde notre approche.

Exemple 7.9.
Considérons le jeu de données du tableau 7.1, et l'ensemble de mesures M=$\{m_1$:$\texttt{freq}, m_2$:$\texttt{émergence}_\text{j}, m_3$:$\texttt{aire}, m_4$:$\texttt{mean}\}$. Les edge-skypatterns par rapport à M sont décrits colonne 1 du tableau 7.3. On peut vérifier que *Edge-Skypattern*(M) est bien un sur-ensemble de chaque nœud du cube (cf. le tableau 7.2).

7.3.1b Cheminement

Soient $M_u \subseteq M$ et $m \in M \setminus M_u$, le théorème 5 a permis de caractériser un type particulier de motifs, à savoir les motifs qui sont des skypatterns pour un nœud fils, mais qui ne sont pas des skypatterns pour un nœud père. Mais, de tels skypatterns sont des edge-skypatterns pour le nœud fils (cf. le théorème 1). Ils seront donc aussi des edge-skypatterns pour le nœud père. En effet, le théorème 8 démontre que l'opérateur *Edge-Sky* est monotone. Enfin, le théorème 9 fournit le résultat final.

Théorème 8 *Monotonie de l'opérateur Edge-Sky.* Soient M un ensemble de mesures, et $M_u \subseteq M$, alors $\forall m \in M \setminus M_u$, *Edge-Skypattern*$(M_u) \subseteq$ *Edge-Skypattern*$(M_u \cup \{m\})$.

Par contradiction. Soient $\texttt{x} \in$ *Edge-Skypattern*(M_u) et supposons que $\texttt{x} \notin$ *Edge-Skypattern*$(M_u \cup \{m\})$. Alors, $\exists \texttt{y} \neq x$ tel que $\texttt{y} \gg_{M_u \cup \{m\}} \texttt{x}$. On en déduit : (1) $\forall m' \in M_u, m'(\texttt{y}) > m'(\texttt{x})$ et (2) $m(\texttt{y}) > m(\texttt{x})$. Selon (1), y domine strictement x par rapport à M_u. Ce qui contredit : $\texttt{x} \in$ *Edge-Skypattern*(M_u). □

Théorème 9 *Résultat principal.* Soit M un ensemble de mesures.

$$\forall M_u \subseteq M, Skypattern(M_u) \subseteq Edge\text{-}Skypattern(M)$$

Démonstration. Soit $M_u \subseteq M$, $Skypattern(M_u) \subseteq Edge\text{-}Skypattern(M_u)$ (par le théorème 1). Par le théorème 8, $Edge\text{-}Skypattern(M_u) \subseteq Edge\text{-}Skypattern(M)$. D'où le résultat. □

7.3.2 Calcul de $Edge\text{-}Skypattern(M)$

$Edge\text{-}Skypattern(M)$ peut être calculé de deux manières différentes :
- à l'aide des CSP dynamiques, comme le montre la section 6.2.1 (cf. page 88).
- à l'aide de EDGE-AETHERIS, qui est une extension de AETHERIS permettant d'extraire les edge-skypatterns par rapport à un ensemble de mesures M (cf. la section 6.4 à la page 95)).

7.3.3 Calcul du cube de skypatterns

Cette section montre comment le problème de calcul d'un cube de skypatterns (par rapport à un ensemble de mesures M) peut être transformé en un problème équivalent de calcul d'un cube de skylines (en $|M|$ dimensions).

7.3.3a D'un cube à l'autre

Soient M un ensemble de mesures et $k = |M|$ et f une fonction de $\mathcal{L}_\mathcal{I}$ dans $\mathbb{R}^k$ qui associe, à chaque motif $\mathtt{x} \in \mathcal{L}_\mathcal{I}$, un point $f(\mathtt{x}) \in \mathbb{R}^k$ de coordonnées $(m_1(\mathtt{x}), m_2(\mathtt{x}), \ldots, m_k(\mathtt{x}))$. Soit $P = \{f(\mathtt{x}) \mid \mathtt{x} \in \mathcal{L}_\mathcal{I}\}$, P est un multi-ensemble : en effet, soit $\mathtt{y}', \mathtt{y}'' \in \mathcal{L}_\mathcal{I}$ tel que $\mathtt{y}' \neq \mathtt{y}''$. Si $\mathtt{y}'$ et $\mathtt{y}''$ sont indistincts par rapport à M, alors $f(\mathtt{y}') = f(\mathtt{y}'')$.

Exemple 7.10.
Le tableau 7.3 montre la correspondance entre $Edge\text{-}Skypattern(M)$ et les points de $\mathbb{R}^4$ ($|M| = 4$). Ainsi, $f(B)$ est le point de coordonnées $(6, 0.66, 6, 40)$, et $f(BCD)$ est le point de coordonnées $(5, 0.88, 15, 25)$.

Soient $M_u \subseteq M$ et $Skyline(M_u)$ l'ensemble des points skyline (de P) par rapport à M_u. La propriété suivante établit le lien entre ces deux ensembles, et donc le lien entre un cube de skypatterns et le cube de skylines qui lui est associé (selon f).

Théorème 10 . Soit M un ensemble de mesures, $\forall M_u \subseteq M$,

$$Skypattern(M_u) = \{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid f(\mathtt{x}) \in Skyline(M_u)\}$$

Par contradiction. Soit $M_u \subseteq M$, supposons que il existe $\mathtt{x} \in \mathcal{L}_\mathcal{I}$ tel que $\mathtt{x} \in Skypattern(M_u)$ et $f(\mathtt{x}) \notin Skyline(M_u)$. D'où :

$$\begin{array}{l} \mathtt{x} \in Skypattern(M_u) \\ \nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I}, \quad \mathtt{y} \succ_{M_u} \mathtt{x} \\ \nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I}, \quad f(\mathtt{y}) \succ_{M_u} f(\mathtt{x}) \\ f(\mathtt{x}) \in Skyline(M_u) \end{array}$$

Conduisant à une contradiction. □

Par conséquent, les classes d'équivalence pour les skypatterns (i.e. $\mathbf{x}$) peuvent être déduites des classes d'équivalence pour les skylines (i.e. $f(\mathbf{x})$), ceci afin d'obtenir directement la représentation concise du cube de skypatterns.

Exemple 7.11.
Soit $M_u = \{m_1, m_3\}$, en utilisant l'extracteur de skylines et la fonction f, on obtient :

$$Skyline(M_u) = \{(\mathbf{5}, 0.88, \mathbf{15}, 25), (\mathbf{4}, 1.33, \mathbf{16}, 40), (\mathbf{6}, 0.66, \mathbf{6}, 40), (\mathbf{6}, 1.33, \mathbf{8}, 55)\}$$

$Skypattern(M_u) = \{BCD, BCDE, B, E\}$ en utilisant de nouveau f (cf. le tableau 7.3).

7.3.3b Utilisation d'un majorant

Soient M un ensemble de mesures, et E un sur-ensemble de tous les ensembles de skypatterns du cube associé à M, et soit $f(E)$ l'ensemble de points de $\mathbb{R}^{|M|}$ associés. Appliquer un extracteur de cube de skylines sur $f(E)$ permet de déterminer le cube de skypatterns associé à M (cf. le théorème 10).

Plusieurs sur-ensembles E peuvent être envisagés :

- Choisir $E = \mathcal{L}_\mathcal{I}$ constituerait une approche naïve car $f(\mathcal{L}_\mathcal{I})$ contiendrait $2^{|\mathcal{I}|}$ points.
- Choisir E comme étant l'ensemble des fermés par rapport à M' (où M' est l'ensemble de mesures "skylineables" de M) constituerait une meilleure approche, mais le nombre de motifs fermés peut s'avérer élevé (cf. la section 7.4.2-δ).
- Enfin, $E = Edge\text{-}Skypattern(M)$ est un sur-ensemble de très bonne qualité (pas trop grand). En effet, les edge-skypatterns se situent sur la frontière Pareto, tandis que les skypatterns sont les sommets de cette frontière. Il nous semble difficile de trouver une meilleure approximation qui soit aussi facile à calculer.

7.4 Étude de cas : Mutagénicité

Dans cette section, nous présentons une évaluation expérimentale sur un jeu de données réel de grande taille, extrait à partir des données de mutagénicité [Hansen *et al.*, 2009] (un problème majeur dans l'évaluation des risques des substances chimiques) fourni par le CERMN. Ce jeu de données contient $|\mathcal{T}| = 6512$ molécules. Les molécules sont représentées en utilisant $|\mathcal{I}| = 1073$ sous-graphes fréquents fermés initialement extraits de $\mathcal{T}$ [39] avec un seuil de fréquence de 2%. Pour étudier le caractère mutagène des substences chimiques, les chimistes utilisent jusqu'à $|M| = 11$ mesures : cinq d'entre elles sont de type fouille (fréquence, taux de croissance, etc) et permettent d'exprimer différents types de connaissances de base. Les six autres mesures sont liées aux propriétés topologiques, géométriques et chimiques des molécules.

La mise en œuvre des méthodes fondées sur les CSP dynamiques ont été réalisées en *Gecode* (cf. l'annexe A.1) et nous avons choisi ORION comme extracter de cube de skylines (cf. la section 4.4). Toutes les expérimentations ont été menées sur un ordinateur exécutant le système d'exploitation Linux avec un processeur Core i3 à 2.13 GHz et une mémoire vive de 4 Go.

7.4.1 Analyse du temps de calcul

Nous avons comparé 6 méthodes pour le calcul du cube de skypatterns :

39. Une molécule Ch contient un item A si Ch supporte A et A est un sous-graphe fréquent fermé de $\mathcal{T}$.

— deux méthodes dites "base-line" :
  — BASE-LINE-AETHERIS, qui applique AETHERIS aux $2^{|M|} - 1$ sous-ensembles non-vides de M,
  — BASE-LINE-CP+SKY, qui applique CP+SKY aux $2^{|M|} - 1$ sous-ensembles non-vides de M,

— l'approche bottom-up : CP+SKY+CUBE (cf. la section 7.2),

— deux approches par relaxation (cf. la section 7.3) :
  — CP+EDGE-SKY +ORION, qui calcule l'ensemble des edge-skypatterns par rapport à M à l'aide de CP+EDGE-SKY, puis applique ORION pour calculer le cube de skylines selon M.
  — EDGE-AETHERIS +ORION, qui calcule l'ensemble des edge-skypatterns par rapport à M à l'aide de EDGE-AETHERIS, puis applique ORION pour calculer le cube de skylines selon M.

— MICMAC +ORION, qui calcule l'ensemble des motifs fermés par rapport à M' (où M' est l'ensemble de mesures "skylineables" de M) à l'aide de MICMAC, puis applique ORION pour calculer le cube de skylines selon M.

Pour les deux méthodes "base-line", le temps de calcul correspond au cumul des temps de calcul requis pour chaque sous-ensemble non vide de M. Pour les deux approches par relaxation et MICMAC +ORION, le temps de calcul est la somme des temps de calcul des deux étapes : tout d'abord, calcul de l'approximation (soit par les motifs fermés ou les edge-kypatterns), puis calcul du cube en utilisant ORION.

La figure 7.5 compare les temps de calcul des six méthodes en fonction du nombre de mesures $|M|$. L'échelle est logarithmique. Le tableau 7.4 analyse de manière détaillée les temps de calcul des 6 méthodes. Pour chaque méthode, et pour un nombre de k mesures ($|M| = k$), le temps de calcul indiqué est la moyenne des temps de calcul de tous les $\binom{11}{k}$ cubes de skypatterns possibles.

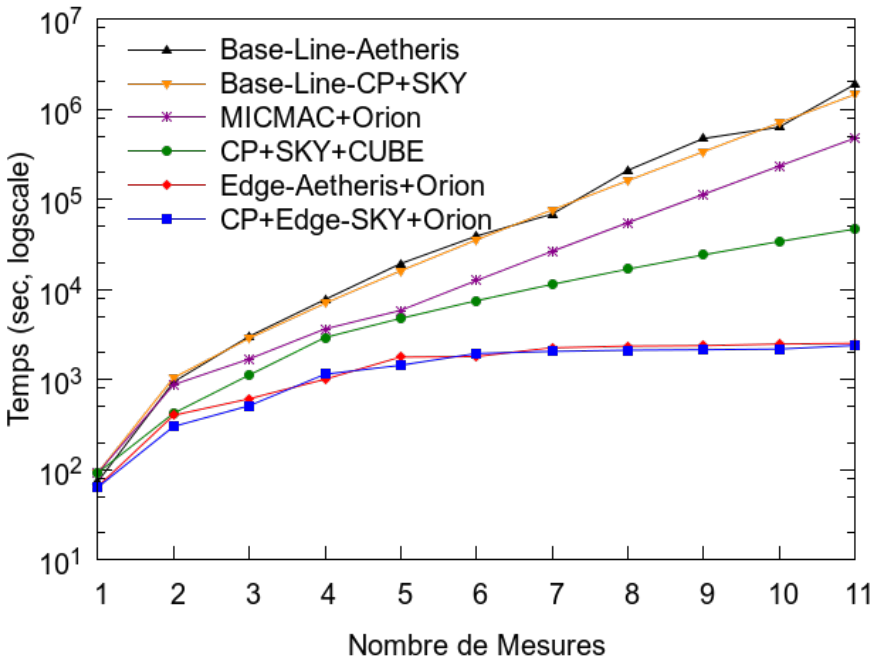


FIGURE 7.5 – Comparaison des temps de calcul pour les 6 méthodes.

Les deux méthodes "base-line" ont un comportement similaire car les performances d'AETHERIS et CP+SKY sont comparables (cf. la section 6.4.1). Comme attendu, les méthodes "base-line" sont, de très loin, les moins performantes par rapport aux quatre autres méthodes. Comparé à CP+EDGE-SKY +ORION (resp. EDGE-AETHERIS), pour un petit nombre de mesures ($2 \leq k \leq 4$), l'accélération est d'environ 5.15

(resp. 4.86) en moyenne (cf. les colonnes $2-3$ et $6-7$ du tableau 7.4b). Pour $k=9$, il y a deux ordres de grandeur (accélération >100). Pour $k=11$, CP+Edge-Sky +Orion (resp. Edge-Aetheris) nécessite 39 minutes (resp. 42 minutes) pour calculer le cube, alors que les deux méthodes "base-line" ont besoin de de plus de 400h (accélération moyenne de 659).

Les temps de calcul obtenus par MICMAC +Orion sont de qualité très moyenne comparés à ceux obtenus par les deux méthodes par relaxation. Pour $k=7$, il y a un ordre de grandeur (accélération d'environ 12) (cf. les colonnes 4 et 8 du tableau 7.4b). Pour $k=11$, l'accélération atteint presque deux ordres de grandeur. Ces résultats peuvent s'expliquer par le fait que l'approximation par les motifs fermés reste trop grossière par rapport à l'approximation par les edge-skypatterns, et génère un nombre trop élevé de points (cf. la section 7.4.2-δ).

Les deux méthodes par relaxation (Edge-Aetheris +Orion et CP+Edge-Sky +Orion) ont un comportement similaire et surpassent clairement l'approche bottom-up (CP+Sky+Cube). En outre, plus le nombre de mesures augmente, plus l'accélération est plus importante. En effet, chaque fois qu'une nouvelle mesure est ajoutée à M, le nombre de nœuds à considérer est deux fois plus grand et les accélérations sont multipliées par un facteur entre 1.25 et 1.45 (cf. les colonnes 5 et 9 du tableau 7.4b). Notons toutefois, les bonne performances de l'approche bottom-up CP+Sky+Cube, qui se classe en troisième position.

$\|M\|$	(2)	(3)	(4)	(5)	(6)	(7)
2	15m42s	17m27s	14m39s	7m03s	6m41s	**5m01s**
3	50m19s	47m55s	28m11s	18m44s	10m13s	**8m33s**
4	2h08m40s	1h56m58s	1h00m44s	48m43s	**16m47s**	19m06s
5	5h21m47s	4h28m09s	1h37m39s	1h19m30s	29m42s	**24m04s**
6	10h49m45s	9h50m41s	3h29m20s	2h04m45s	**30m06s**	32m28s
7	19h01m22s	21h08m11s	7h20m13s	3h09m34s	37m37s	**34m14s**
8	58h05m32s	44h41m11s	15h15m33s	4h40m02s	39m14s	**35m21s**
9	131h03m16s	93h36m37s	31h31m17s	6h43m06s	39m44s	**35m45s**
10	175h17m57s	194h46m36s	64h44m06s	9h26m41s	41m25s	**36m24s**
11	523h11m58s	402h27m40s	131h58m47s	12h59m35s	42m07s	**39m52s**

(2) Base-Line-Aetheris (4) MICMAC +Orion (6) Edge-Aetheris +Orion
(3) Base-Line-CP+Sky (5) CP+Sky+Cube (7) CP+Edge-Sky +Orion

(a) Temps de calcul

$\|M\|$	$\frac{(2)}{(6)}$	$\frac{(3)}{(6)}$	$\frac{(4)}{(6)}$	$\frac{(5)}{(6)}$	$\frac{(2)}{(7)}$	$\frac{(3)}{(7)}$	$\frac{(4)}{(7)}$	$\frac{(5)}{(7)}$
2	2.35	2.61	2.19	1.05	3.12	3.47	2.91	1.40
3	4.92	4.69	2.76	1.83	5.88	5.60	3.30	2.19
4	7.66	6.97	3.62	2.90	6.74	6.12	3.18	2.55
5	10.83	9.03	3.29	2.68	13.37	11.14	4.06	3.30
6	21.58	19.61	6.95	4.14	20.00	18.18	6.44	3.84
7	30.34	33.71	11.70	5.04	33.34	37.04	12.86	5.54
8	88.80	68.31	23.33	7.13	98.59	75.84	25.90	7.92
9	197.88	141.34	47.60	10.14	219.91	157.08	52.89	11.27
10	253.95	282.16	93.78	13.68	288.83	320.92	106.66	15.56
11	745.19	573.22	187.98	18.51	787.42	605.71	198.63	19.56

(b) Accélération

Table 7.4 – Analyse du temps de calcul (Mutagénicité).

Enfin, nous avons également mesuré, pour les deux méthodes par relaxation, les temps de calcul nécessaires pour calculer *Edge-Skypattern*(M) et les temps de calcul requis par Orion. Pour $8 \leq k \leq 11$, le calcul de *Edge-Skypattern*(M) représente environ 90% du temps de calcul total (cf. le tableau 7.5). Cela montre, une fois de plus, l'importance de la qualité de notre approximation. Toutefois, la répartition peut sembler disproportionnée, mais calculer le cube de skypatterns est beaucoup plus difficile que calculer le cube de skylines (cf. la section 6.1.2).

$\|M\|$	CP+Edge-Sky	Orion
6	30m14s	2m14s
7	31m44s	2m29s
8	32m56s	2m24s
9	33m12s	2m32s
10	33m36s	2m48s
11	37m04s	2m47s

Table 7.5 – Temps désagrèges pour CP+Edge-Sky +Orion

7.4.2 Analyse qualitative de nos méthodes

Cette section présente une analyse qualitative de nos trois méthodes (une bottom-up : CP+Sky+Cube et deux approximatives : Edge-Aetheris +Orion et CP+Edge-Sky +Orion) selon les quatre critères suivants :

i) Qualité de notre représentation concise,

ii) Qualité de nos règles de dérivation,

iii) Efficacité de notre condition suffisante,

iv) Qualité de l'approximation.

Les trois premiers critères concernent uniquement l'approche bottum-up, alors que le quatrième critère s'applique pour les trois méthodes. Pour ce dernier critère, nous considérons, pour un ensemble de mesures M :

— le nombre de skypatterns (distincts) du cube :
$nCube(M) = |\cup_{M_u \subseteq M, M_u \neq \emptyset} Skypattern(M_u)|$,

— le nombre de motifs fermés par rapport par rapport à M' : $nClosed(M')$,

— le nombre d'edge-skypatterns selon M :
$nEdge(M) = |Edge\text{-}Skypattern(M)|$.

Pour évaluer l'efficacité de notre approximation, nous déterminons les "surplus" de motifs extraits par chacune des trois méthodes. Pour MICMAC +Orion, les "surplus" sont mesurés par la proportion de motifs fermés qui ne sont pas des skypatterns (pour tout noeud du cube). Pour Edge-Aetheris +Orion et CP+Edge-Sky +Orion, les "surplus" sont mesurés par la proportion d'edge-skypatterns qui ne sont pas des skypatterns (pour tout noeud du cube).

7.4.2-α Qualité de notre représentation concise. La colonne 1 du tableau 7.6 donne le nombre de mesures. La colonne 2 indique le nombre de classes d'équivalence. La colonne 3 indique le rapport entre le nombre de classes d'équivalence et le nombre total de sous-ensembles de mesures.

La colonne 4 (resp. 5) rapporte le nombre total de skypatterns pour la représentation concise (resp. usuelle) (cf. la section 7.2.3) et la colonne 6 donne leur rapport ($\frac{(4)}{(5)}$). Pour $|M| = k$, les valeurs indiquées dans les colonnes (2), (4) et (5) sont les moyennes sur tous les $\binom{11}{k}$ cubes de skypatterns possibles.

Notre représentation concise du cube de skypatterns fournit un résumé et une compression importante des ensembles de skypatterns. Par exemple, pour $|M| = 11$, il y a 401 classes d'équivalence et un nombre total de $15,261$ skypatterns pour la représentation concise. Pour la représentation usuelle, il existe $2,047$ sous-ensembles de mesures et un nombre total de $87,374$ skypatterns, ce qui représente un gain très substantiel supérieur à 80%.

$\|M\|$	(2)	$\frac{(2)}{2^{\|M\|}-1}$	(4)	(5)	$\frac{(4)}{(5)}$
1	1	1.00	338	338	1.00
2	2	0.87	680	753	0.90
3	5	0.75	1,036	1,280	0.80
4	9	0.64	1,421	1,983	0.71
5	16	0.53	1,865	2,982	0.62
6	28	0.44	2,424	4,526	0.53
7	45	0.36	3,200	7,146	0.45
8	73	0.29	4,386	12,015	0.36
9	117	0.23	6,327	21,773	0.23
10	213	0.20	9,619	42,386	0.23
11	401	0.20	15,261	87,374	0.17

(2) Nombre de classes d'équivalence
(4) Nombre de skypatterns pour la representation concise
(5) $\sum_{M_u \subseteq M} |Skypattern(M_u)|$

TABLE 7.6 – Qualité de notre représentation concise (Mutagénicité).

7.4.2-β Qualité de nos règles de dérivation. Afin d'évaluer la qualité des deux règles de dérivation (cf. la section 7.2.1), nous avons mesuré le pourcentage de skypatterns dérivés (vs le nombre total de skypatterns) à chaque niveau du cube. Les valeurs indiquées par la figure 7.6 sont les moyennes sur les 11 cubes possibles de 10 mesures. Pour chaque niveau i ($2 \leq i \leq 10$), la proportion de skypatterns incomparables et indistincts est également indiquée.

Nos règles de dérivation sont très efficaces car elles permettent de déduire environ $80 - 90\%$ des skypatterns, sauf pour les premiers niveaux. En outre, lorsque le nombre de mesures augmente, le nombre de skypatterns indistincts diminue (en pourcentage), alors que le nombre de skypatterns incomparables augmente. En effet, les skypatterns incomparables des nœuds fils restent incomparables pour un nœud père (cf. le théorème 3), alors que les skypatterns indistincts peuvent devenir soit des skypattern incomparables pour un nœud père, demeurer des skypatterns indistincts pour un nœud père, ou ne plus être des skypatterns (cf. le théorème 5).

7.4.2-γ Efficacité de notre condition suffisante. Le théorème 6 fournit une condition suffisante pour éviter de lancer inutilement le calcul à la volée des skypatterns non-derivables. Pour évaluer la portée de cette condition, nous avons mesuré le pourcentage de réussite à chaque niveau d'un cube. Les valeurs indiquées dans le tableau 7.7 sont des moyennes sur les 11 cubes possibles de 10 mesures.

Niveau	2	3	4	5	6	7	8	9	10
Nombre de M_u où le théorème 3 s'applique	6.55	54.55	127.91	183.27	171.82	106.91	42.55	9.82	1.00
Nombre de M_u où le théorème 3 devrait s'appliquer	29.45	102.55	194.73	252.00	210.00	120.00	45.00	10.00	1.00
Taux de succès : (2)/(3)	0.22	0.53	0.66	0.73	0.82	0.89	0.95	0.98	1.00

TABLE 7.7 – Efficacité de notre condition suffisante (Théorème 6).

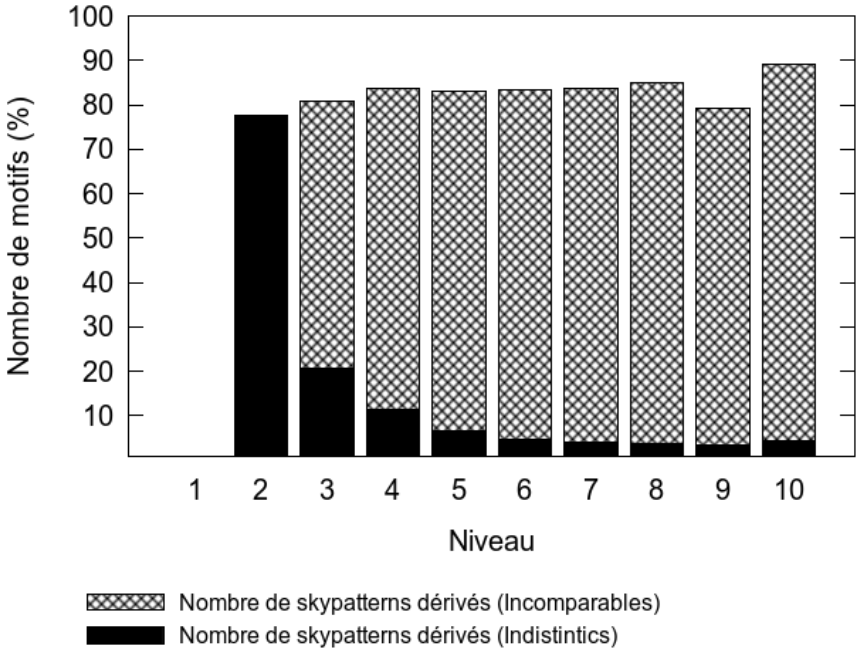


FIGURE 7.6 – Qualité de nos règles de dérivation.

La ligne 1 fournit le niveau dans le treillis (i.e. le nombre de mesures). La ligne 2 indique le nombre de nœuds où notre condition s'applique, alors que la ligne 3 indique le nombre de nœuds où notre condition devrait s'appliquer. La ligne 4 représente le taux de succès. Plus le nombre de mesures augmente, plus notre condition suffisante devient efficace. À partir du niveau 5 et jusqu'au niveau 10, le taux de succès augmente de 73% à 100%. En effet, comme les skypatterns non-dérivables ne peuvent provenir que de skypatterns indistincts, la diminution (en pourcentage) du nombre de skypatterns indistincts lorsque le nombre de mesures augmente (cf. la figure 7.6) explique probablement l'augmentation du taux de succès.

7.4.2-δ Qualité de l'approximation. Pour évaluer la qualité de notre approximation du cube de skypatterns par *Edge-Skypattern*(M), la colonne 5 (resp. 6) du tableau 7.8 rapporte, pour différentes valeurs de $|M|$, le ratio d'edge-skypatterns (resp. motifs fermés) qui ne sont pas des skypatterns. Pour $|M| = k$, les valeurs rapportées sont des moyennes sur tous les $\binom{11}{k}$ cubes de skypatterns possibles. Les colonnes 5 et 6 montrent clairement que *Edge-Skypattern*(M) fournit une bien meilleure approximation que l'ensemble de motifs fermés :

i) le nombre de edge-skyppaterns est toujours plus petit que le nombre de motifs fermés,

ii) pour $|M| \geq 6$, le ratio de edge-skypatterns est entre 30% et 50%, alors que pour les motifs fermés ce rapport est toujours supérieur à 75%.

7.5 Expérimentations sur les jeux de données de l'*UCI*

Les expérimentations ont été menées sur 14 différents jeux de données de l'*UCI*, en considérant cinq mesures : $M = \{\texttt{freq}, \texttt{max}, \texttt{aire}, \texttt{mean}, \texttt{émergence}_\texttt{j}\}$. Les mesures portant sur des valeurs numériques, comme `mean`, ont été appliquées sur des valeurs d'attribut générées aléatoirement dans l'intervalle [0..1].

$\|M\|$	$nCube(M)$	$nEdge(M)$	$nClosed(M')$	(5)	(6)
2	158.84	740.56	311,254.41	0.79	0.99
3	317.96	1,263.55	344,641.68	0.75	0.99
4	588.38	2,056.70	362,054.86	0.71	0.99
5	1,454.11	3,497.94	371,127.89	0.58	0.99
6	3,507.12	6,513.64	382,273.50	0.46	0.99
7	8,352.34	13,316.40	394,456.48	0.37	0.97
8	19,537.90	29,079.40	412,000.00	0.33	0.95
9	44,552.70	65,687.70	437,734.00	0.32	0.89
10	98,515.50	150,060.00	455,986.55	0.34	0.78
11	110,689.00	172,447.00	483,320.00	0.36	0.77

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M')}$

TABLE 7.8 – Qualité de l'approximation (Mutagénicité).

7.5.1 Analyse du temps de calcul

Comme pour le jeu de données Mutagénicité (cf. la section 7.4), nous comparons l'approche bottom-up (CP+SKY+CUBE) et les approches par relaxation (EDGE-AETHERIS +ORION et CP+EDGE-SKY +ORION), avec trois autres méthodes pour calculer les cubes de skypatterns.

1. BASE-LINE-AETHERIS applique AETHERIS aux $2^{|M|} - 1$ sous-ensembles non-vides de M.
2. BASE-LINE-CP+SKY applique CP+SKY aux $2^{|M|} - 1$ sous-ensembles non-vides de M.
3. MICMAC +ORION calcule tous les motif fermés rapport à M' (grâce à MICMAC) et applique ORION pour calculer le cube de skylines.

Jeu de données	$\|\mathcal{I}\|$	$\|\mathcal{T}\|$	densité	(1)	(2)	(3)	(4)	(5)	(6)
austral	55	690	0.272	6m04s	4m15s	6m00s	1m31s	2m18s	**36s**
cleve	43	303	0.325	1m53s	1m21s	28s	21s	18s	**10s**
cmc	28	1,473	0.357	26s	2m23s	31s	22s	**10s**	22s
crx	59	690	0.269	8m40s	5m37s	1m49s	1m13s	1m49s	**40s**
german	76	1,000	0.276	2h34m18s	53m29s	1h33m19s	14m03s	1h16m11s	**10m29s**
heart	38	270	0.368	1m46s	58s	38s	19s	29s	**15s**
horse	75	300	0.235	10m34s	3m32s	3m43s	58s	2m28s	**41s**
hypo	47	3,163	0.389	6h13m57s	51m46s	1h11m19s	44m41s	1h16m19s	**38m58s**
lymph	59	142	0.322	4m32s	49s	1m34s	31s	34s	**19s**
mushroom	119	8,124	0.193	1h53m39s	9h23m28s	55m23s	8h54m43s	**45m51s**	2h00m77s
tic-tac-toe	29	958	0.344	1m10s	2m48s	53s	41s	41s	**16s**
vehicle	58	846	0.327	34m01s	16m41s	5m45s	2m55s	3m32s	**1m42s**
wine	45	178	0.311	1m00s	31s	44s	13s	19s	**5s**
zoo	43	101	0.394	19s	8s	3s	3s	**2s**	**2s**

(1) BASE-LINE-AETHERIS (3) MICMAC +ORION (5) EDGE-AETHERIS +ORION
(2) BASE-LINE-CP+SKY (4) CP+SKY+CUBE (6) CP+EDGE-SKY +ORION

TABLE 7.9 – Temps de calcul pour les différentes méthodes sur les jeux de données *UCI*.

Le tableau 7.9 compare les temps de calcul pour construire le cube de skypatterns associé à M, ceci pour les six méthodes, et pour chaque jeu de données. Les résultats obtenus confirment les tendances constatées pour le jeu de données Mutagénicité (cf. la section 7.4.1) :

i) les deux méthodes base-line ont un comportement similaire, mais sont loin des quatre autres méthodes,

ii) MICMAC +ORION est de qualité très moyenne par rapport aux trois approches non base-line,

iii) les deux méthodes fondées sur la relaxation surpassent l'approche bottom-up CP+SKY+CUBE,

iv) ces deux méthodes sont très proches l'une de l'autre, avec, malgré tout, un léger avantage à CP+Edge-Sky +Orion par rapport à Edge-Aetheris +Orion.

Le jeu de données mushroom constitue une exception notable aux constations que nous venons de faire, car les méthodes à base de fermés (MICMAC +Orion et Edge-Aetheris +Orion) surclassent nettement CP+Sky+Cube et CP+Edge-Sky +Orion. Cela est dû à la faible densité du jeu de données mushroom (environ 19%) et à son petit nombre de motifs fermés par rapport à sa taille.

Enfin, les gains obtenus sont inférieurs à ceux obtenus pour le jeu de données Mutagénicité car $|M|$ est petit. En effet, comme nous pouvions nous y attendre, et comme nous l'avons constaté à la section 7.4.1, plus le nombre de mesures augmente, plus les méthodes base-line deviennent inopérantes.

7.5.2 Analyse qualitative de nos méthodes

7.5.2a Qualité de nos règles de dérivation

Le tableau 7.10 décrit, pour chaque jeu de données, le pourcentage de skypatterns dérivés. Les résultats indiqués sont des moyennes sur les niveaux i ($2 \leq i \leq 5$) du cube. Ces résultats montrent la qualité de nos règles de dérivation qui sont en mesure de déduire (en moyenne) plus de 80% des skypatterns d'un père à partir des skypatterns de ses fils.

austral	cleve	cmc	crx	german	heart	hepatic	horse	hypo	lymph	tic-tac-toe	vehicle	wine	zoo
0.82	0.97	0.90	0.89	0.88	0.86	0.71	0.82	0.79	0.65	0.84	0.66	0.94	0.87

Table 7.10 – Efficacité des règles de dérivation sur les jeux de données *UCI* avec $|M| = 5$.

7.5.2b Qualité de notre approximation par *Edge-Skypattern*(M)

La colonne 5 (resp. la colonne 6) du tableau 7.11 indique, pour chaque jeu de données, la proportion d'edge-skypatterns (resp. de motifs fermés) qui ne sont pas des skypatterns du cube. Cette valeur caractérise, en quelque sorte, la proportion de motifs calculés inutilement du fait de l'approximation. Ces deux colonnes mettent clairement en évidence la qualité et la pertinence de l'approximation par *Edge-Skypattern*(M). Pour la plupart des jeux de données, la proportion (de motifs inutilement calculés) est inférieure à 5% (parfois moins de 1%). De plus, cette proportion est toujours très petite par rapport à celle issue de l'approximation utilisant les motifs fermés (cf. la section 7.3.3b).

Le tableau 7.12 met l'accent sur les trois jeux de données ayant les cubes de plus grande taille : german, hypo et mushroom. Pour $|M| = k$, les valeurs reportées dans les colonnes (2), (3) et (4) sont les valeurs moyennes sur les $\binom{5}{k}$ cubes possibles de skypatterns. Les colonnes 5 et 6 confirment les résultats décrits au tableau 7.11. La proportion de motifs inutilement calculés est toujours inférieure à 1%.

7.6 Conclusion

Tout d'abord, nous avons proposé une méthode bottom-up efficace pour calculer les cubes de skypatterns. Nos règles de dérivation sont en mesure de collecter une grande partie de skypatterns d'un nœud

Jeu de données	$nCube(M)$	$nEdge(M)$	$nClosed(M')$	(5)	(6)
austral	100,534.00	107,632.00	337,399.00	0.07	0.68
cleve	37,021.00	37,105.00	99,840.00	0.00	0,63
cmc	11,744.00	11,791.00	26,862.00	0.00	0.56
crx	102,690.00	104,377.00	409,615.00	0.02	0.75
german	1,138,533.00	1,149,315.00	4,715,852.00	0.01	0.76
heart	33,355.00	34,269.00	91,020.00	0.03	0.62
horse	119,446.00	119,587.00	297,627.00	0.00	0.60
hypo	671,929.00	677,051.00	1,605,518.00	0.01	0.58
lymph	45,799.00	54,880.00	161,447.00	0.17	0.66
mushroom	554,943.00	555,377.00	1,708,099.00	0.00	0.67
tic-tac-toe	15,020.00	15,310.00	49,822.00	0.02	0.69
vehicle	126,322.00	127,682.00	805,311.00	0.01	0.84
wine	13,835.00	13,941.00	47,846.00	0.01	0.71
zoo	4,375.00	4,509.00	16,697.00	0.03	0.73

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M')}$

TABLE 7.11 – Qualité de l'approximation sur les jeux de données *UCI*.

$\|M\|$	$nCube(M)$	$nEdge(M)$	$nClosed(M')$	(5)	(6)
			german		
2	315,280.00	315,646.00	3,942,910.00	0.00	0.92
3	556,458.00	558,325.00	4,242,910.00	0.00	0.87
4	835,367.00	840,539.00	4,500,560.00	0.01	0.81
5	1,138,533.00	1,149,315.00	4,715,852.00	0.01	0.76
			hypo		
2	245,748.00	246,882.00	1,267,060.00	0.00	0.81
3	380,205.00	383,674.00	1,453,550.00	0.01	0.74
4	522,325.00	527,288.00	1,566,360.00	0.01	0.66
5	671,929.00	677,051.00	1,605,518.00	0.01	0.58
			mushroom		
2	108,658.00	108,675.00	920,010.00	0.00	0.88
3	236,782.00	236,860.00	1,216,090.00	0.00	0.81
4	391,253.00	391,465.00	1,478,786.00	0.00	0.74
5	554,943.00	555,377.00	1,708,099.00	0.00	0.67

(5) $1 - \frac{nCube(M)}{nEdge(M)}$ (6) $1 - \frac{nCube(M)}{nClosed(M')}$

TABLE 7.12 – Zoom sur les trois jeux de données *UCI* sélectionnés.

père. Les skypatterns non-dérivables sont calculés à la volée grâce aux CSP dynamiques. La stratégie bottom-up facilite la construction d'une représentation concise du cube en fonction des classes d'équivalence des skypatterns.

Puis, nous avons proposé une seconde méthode fondée sur la relaxation des skypatterns et utilisant les edge-skypatterns. Cette méthode repose sur le fait que tout ensemble de skypatterns d'un nœud est inclus dans l'ensemble des edge-skypatterns de M. Le problème de construction du cube de skypatterns peut alors être facilement ramené au calcul d'un cube de skylines en $|M|$ dimensions. Ce calcul est effectué par un extracteur existant de cubes de skylines.

Les expérimentations menées montrent que la seconde approche s'avère plus performante que la première et permet de construire des cubes de skypatterns pour de plus grands ensembles de mesures. Enfin,

la navigation dans le cube est une perspective très prometteuse, les classes d'équivalence permettant déjà de donner l'importance des mesures ayant le même ensemble de skypatterns.

"À mon avis, aucune conception est susceptible d'être optimale pour tout le monde."

Donald Norman

Chapitre 8

Motifs optimaux

Sommaire

Ce chapitre introduit la notion de motifs optimaux (MO) qui sont les meilleurs motifs selon une préférence définie par l'utilisateur. Tout d'abord, nous montrons que les MO permettent d'exprimer de nombreux problèmes d'extraction d'ensembles de motifs : libres, fermés, maximaux, skypatterns, top-k, motifs pics, miki, sous-groupes pertinents, ... Puis, nous proposons une méthode générique, reposant sur les CSP dynamiques, permettant d'extraire n'importe quelle sorte de MO. L'idée majeure est la suivante : dès qu'une solution est obtenue, une contrainte est ajoutée dynamiquement au CSP courant afin de rechercher une solution de meilleure qualité et ainsi réduire l'espace de recherche. Le processus s'arrête lorsqu'aucune meilleure solution ne peut être obtenue. Nous montrons alors que chaque sorte de MO peut être caractérisée par une contrainte initiale et un ensemble de contraintes ajoutées dynamiquement. Nous pensons que la déclarativité et la généricité de notre approche ouvrent la voie à la définition et à la découverte de nouveaux ensembles de motifs.

La section 8.1 définit la notion de MO, et montre que les MO permettent d'exprimer de nombreux problèmes d'extraction d'ensembles de motifs. La section 8.2 présente notre approche générique pour extraire les MO. La section 8.3 synthétise les travaux relatifs. La section 8.4 décrit l'étude expérimentale menée sur plusieurs sortes de MO et discute la qualité des performances obtenues par rapport aux méthodes ad hoc.

8.1 Contexte

Les Motifs Optimaux (MO) permettent de caractériser les problèmes d'extraction de motifs dont les solutions sont optimales pour une préférence donnée. Cette section définit formellement la notion de MO et montre que de nombreux problèmes d'extraction d'ensembles de motif peuvent se modéliser sous forme de MO : libres, fermés, maximaux, skypatterns, top-k, motifs pics, miki, sous-groupes pertinents, ...

8.1.1 Définitions

Soient $\mathcal{I}$ un ensemble d'items, $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \emptyset$ le langage des motifs associés, et reprenons la définition 1.24 d'une relation préférence introduite à la page 21.

Définition 8.1. Préférence
Une **préférence** $\rhd$ est une relation d'ordre strict [40] et partiel [41] sur $\mathcal{L}_\mathcal{I}$. Soient $\mathtt{x}$ et $\mathtt{y}$ deux motifs, $\mathtt{x} \rhd \mathtt{y}$ indique que $\mathtt{x}$ est strictement préféré à $\mathtt{y}$.

Exemple 8.1.

- Soit m une **mesure**, $\mathtt{x} \rhd \mathtt{y} = m(\mathtt{x}) > m(\mathtt{y})$.
- **La Pareto-dominance** (cf. la définition 6.1) : Soit M un ensemble de mesures, un motif $\mathtt{x}$ domine un autre motif $\mathtt{y}$ par rapport à M (noté par $\mathtt{x} \succ_M \mathtt{y}$) si et seulement si $\forall m \in M, m(\mathtt{x}) \geq m(\mathtt{y})$ et $\exists m' \in M, m'(\mathtt{x}) > m'(\mathtt{y})$. Ainsi, $\mathtt{x} \rhd \mathtt{y} = \mathtt{x} \succ_M \mathtt{y}$

Définition 8.2. Motif Optimal (MO)
Étant donné une préférence $\rhd$, un motif $\mathtt{x}$ est optimal ssi $\nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_n \in \mathcal{L}_\mathcal{I}, \forall 1 \leq i \leq n,\ \mathtt{y}_i \rhd \mathtt{x}$

Ainsi, $\mathtt{x}$ est optimal s'il n'existe pas n motifs qui soient préférés à $\mathtt{x}$. Notre but est d'extraire l'ensemble de MO :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \texttt{élémentaire}(\mathtt{x}) \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_n \in \mathcal{L}_\mathcal{I}, \underbrace{\forall 1 \leq i \leq n,\ \mathtt{y}_i \rhd \mathtt{x}}_{\bigwedge_{1 \leq i \leq n} \mathtt{y}_i \rhd \mathtt{x}}\}$$

La contrainte $\texttt{élémentaire}(\mathtt{x})$ permet de préciser que le motif $\mathtt{x}$ doit satisfaire une propriété de base. Elle peut être vide, mais dans la pratique elle s'avère très utile, les problèmes de fouille demandant souvent que les motifs recherchés vérifient chacun une propriété particulière comme par exemple avoir une fréquence minimale, être un fermé, ...

Cette modélisation, qui peut apparaître simple, est puissante pour une double raison. D'une part, elle est générale et permet de définir de nombreux problèmes d'extraction d'ensembles de motifs (sa force réside dans les comparaisons multiples autorisées entre $\mathtt{x}$ et les $\mathtt{y}_i$). D'autre part, elle est propice à une méthode efficace de résolution fondée sur les CSP dynamiques (cf. la section 8.2).

8.1.2 Exemples

Nous présentons succinctement plusieurs problèmes d'extraction de motifs connus et nous montrons qu'ils sont des MO. Nous ne détaillerons pas ici leurs intérêts respectifs. Pour cela, nous renvoyons le lecteur aux articles définissant ces problèmes. Dans la suite de cette section, $minfr$ désignera un seuil de fréquence quelconque, y compris nul.

40. Une relation binaire est une relation d'ordre strict quand elle est irréflexive et transitive.
41. Il existe deux motifs $\mathtt{x}$ et $\mathtt{y}$ ($\mathtt{x} \neq \mathtt{y}$) qui sont incomparables par rapport à $\rhd$ (i.e. $\mathtt{x} \not\rhd \mathtt{y}$ et $\mathtt{x} \not\lhd \mathtt{y}$).

8.1.2a Motifs fermés [Pasquier *et al.*, 1999a]

Les motifs fermés forment une couverture du jeu de données en sélectionnant uniquement les motifs $\mathtt{x}$ dont aucune spécialisation $\mathtt{y} \supset \mathtt{x}$ a une fréquence égale à celle de $\mathtt{x}$:

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) \geq minfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \supset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})\}$$

8.1.2b Motifs libres [Boulicaut *et al.*, 2003, Bastide *et al.*, 2000]

Un motif $\mathtt{x}$ est libre s'il n'existe aucune généralisation $\mathtt{y} \subset \mathtt{x}$ ayant la même couverture :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) \geq minfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})\}$$

8.1.2c Motifs δ-libres [Boulicaut *et al.*, 2003]

Un motif $\mathtt{x}$ est δ-libre s'il n'existe aucune généralisation $\mathtt{y} \subset \mathtt{x}$ dont l'écart entre la couverture de $\mathtt{y}$ et celle de $\mathtt{x}$ est inférieure ou égale de δ ($\delta \geq 0$) :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) \geq minfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) - \mathtt{freq}(\mathtt{x}) \leq \delta\}$$

8.1.2d Motifs maximaux fréquents ($\mathcal{B}d^+$) [Gunopulos *et al.*, 1997, Bayardo, 1998]

Un motif fréquent est maximal (i.e., appartient $\mathcal{B}d^+$ des fréquents) s'il n'existe aucune spécialisation $\mathtt{y} \supset \mathtt{x}$ qui soit aussi fréquente :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) \geq minfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \supset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) \geq minfr\}$$

8.1.2e Motifs minimaux non-fréquents ($\mathcal{B}d^-$) [Mannila et Toivonen, 1997]

Un motif non-fréquent est minimal (i.e., appartient $\mathcal{B}d^-$ des fréquents) s'il n'existe aucune généralisation $\mathtt{y} \subset \mathtt{x}$ qui soit aussi non-fréquente :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) < maxfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) < maxfr\}$$

8.1.2f Skypatterns [Soulet *et al.*, 2011]

Soit M un ensemble de mesures, un skypattern est un motif non-dominé par rapport à M :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{fermé}_{M'}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \succ_M \mathtt{x}\}$$

La contrainte $\mathtt{fermé}_{M'}(\mathtt{x})$ impose que $\mathtt{x}$ est un motif fermé pour M' pour éviter les skypatterns redondants.

8.1.2g Skypatterns souples [Ugarte *et al.*, 2014d]

Soit M un ensemble de mesures.

Edge-skypatterns Un Edge-skypattern est un motif non-dominé strictement par rapport à M :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{fermé}_{M'}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I} : \mathtt{y} \gg_M \mathtt{x}\}$$

δ-skypatterns Un δ-skypattern est un motif non-δ-dominé par rapport à M :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{fermé}_{M'}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I} : \mathtt{y} \succ^{\delta}_{M} \mathtt{x}\}$$

La contrainte $\mathtt{fermé}_{M'}(\mathtt{x})$ impose que $\mathtt{x}$ est un motif fermé pour M' pour éviter les skypatterns souples redondants.

8.1.2h Motifs pics [Crémilleux et Soulet, 2008]

Soient $d(\mathtt{x},\mathtt{y}) = |\mathtt{x} \setminus \mathtt{y}| + |\mathtt{y} \setminus \mathtt{x}|$ une distance, m une mesure, ε un entier et δ un réel. Un motif pic possède une valeur selon m jugée grande par rapport à celles de ses voisins (au sens de d), alors :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{freq}(\mathtt{x}) \geq 1 \wedge \nexists\, \mathtt{y} \in \mathcal{L}_\mathcal{I} : d(\mathtt{x},\mathtt{y}) \leq \delta \wedge \varepsilon \times \mathtt{m}(\mathtt{y}) > \mathtt{m}(\mathtt{x})\}$$

8.1.2i top-k [Han *et al.*, 2002]

Soient m une mesure, et k un entier. *top-k* est l'ensemble des k meilleurs motifs selon m :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{freq}(\mathtt{x}) \geq 1 \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \mathcal{L}_\mathcal{I} : \underbrace{\forall 1 \leq i \leq k, \mathtt{m}(\mathtt{y}_i) > \mathtt{m}(\mathtt{x})}_{\min\limits_{1 \leq i \leq k} \{\mathtt{m}(\mathtt{y}_i)\} > \mathtt{m}(\mathtt{x})}\}$$

Pour mieux comprendre le top-k, on présente trois cas particuliers donnés du top-k présentés dans la littérature.

8.1.2i-α top-k tuiles [Geerts *et al.*, 2004] Soit k un entier, les *top-k* tuiles (cf. la section 1.6 à la page 31) est l'ensemble des k tuiles avec les plus grandes aires :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{freq}(\mathtt{x}) \geq 1 \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \mathcal{L}_\mathcal{I} : \forall 1 \leq i \leq k, \mathtt{aire}(\mathtt{y}_i) > \mathtt{aire}(\mathtt{x})\}$$

8.1.2i-β top-k motifs sous contraintes souples [Ugarte *et al.*, 2012a, Ugarte *et al.*, 2013b] Soient k un entier, $\mathtt{q}(\mathtt{x})$ une requête souple, $\mathtt{q}'(\mathtt{x})$ la requête dure équivalente associée à $\mathtt{q}(\mathtt{x})$ et $\theta_\mathtt{q}$ une mesure d'intérêt pour les motifs souples (cf. la section 5.2.3 à la page 69). Les top-k motifs souples (cf. la section 5.2 à la page 68) forment l'ensemble des k meilleurs motifs selon la mesure $\theta_\mathtt{q}$:

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{q}'(\mathtt{x}) \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \mathcal{L}_\mathcal{I} : \forall 1 \leq i \leq k, \theta_\mathtt{q}(\mathtt{y}_i) > \theta_\mathtt{q}(\mathtt{x})\}$$

8.1.2i-γ top-k motifs fermés fréquents [Jabbour *et al.*, 2013b] Soient k un entier et ω un seuil de taille, les *top-k* motifs fermés fréquents sont l'ensemble des k motifs fermés avec les plus grandes valeurs de supports :

$$\{\mathtt{x} \in \mathcal{L}_\mathcal{I} \mid \ \mathtt{fermé}(\mathtt{x}) \wedge \mathtt{taille}(\mathtt{x}) \geq \omega \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \mathcal{L}_\mathcal{I} : \forall 1 \leq i \leq k, \mathtt{taille}(\mathtt{y}_i) \geq \omega \wedge \mathtt{freq}(\mathtt{y}_i) > \mathtt{freq}(\mathtt{x})\}$$

8.1.2j The N-most interesting k-itemsets [Cheung et Fu, 2004]

Soient k et N deux entiers, *N-most* est l'ensemble des N plus fréquents k-itemsets :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \mathtt{taille(x)} = k \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_N \in \mathcal{L}_{\mathcal{I}} : \underbrace{\forall 1 \leq i \leq N, \mathtt{freq}(\mathtt{y}_i) > \mathtt{freq(x)}}_{\min\limits_{1 \leq i \leq N} \{\mathtt{freq}(\mathtt{y}_i)\} > \mathtt{freq(x)}}\}$$

8.1.2k Découverte de sous-groupes pertinents [Novak *et al.*, 2009]

Un sous-groupe pertinent rassemble des motifs qui discriminent T_1 de T_2, où T_1 de T_2 sont 2 partitions de classe de $\mathcal{T}$ (e.g. positive ou négative, toxique ou non toxique) :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \mathtt{freq_j(x)} \geq minfr \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : T_1(\mathtt{y}) \supseteq T_1(\mathtt{x}) \wedge T_2(\mathtt{y}) \subseteq T_2(\mathtt{x}) \wedge (T(\mathtt{y}) = T(\mathtt{x}) \implies \mathtt{y} \subset \mathtt{x})\}$$

8.1.2l Pattern compression problem [Xin *et al.*, 2005]

Soient $d(\mathtt{x}, \mathtt{y}) = 1 - \frac{|T(\mathtt{x}) \cap T(\mathtt{y})|}{|T(\mathtt{x}) \cup T(\mathtt{y})|}$ une distance et δ un seuil. Un motif $\mathtt{x}$ est *représentatif* si et seulement si aucun motif $\mathtt{y}$ dans son voisinage ($d(\mathtt{x}, \mathtt{y}) \leq \delta$) n'est inclus dans $\mathtt{x}$ ($\mathtt{y} \not\subset \mathtt{x}$). Le problème de compression de motifs consiste à trouver un ensemble de motifs représentatifs :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \mathtt{fermé(x)} \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : d(\mathtt{x}, \mathtt{y}) \leq \delta \wedge \mathtt{y} \not\subset \mathtt{x}\}$$

8.1.2m Maximally informative k-itemset [Knobbe et Ho, 2006a]

Soient $\mathtt{x}$ un k-itemset (i.e. $\mathtt{taille(x)} = k$) et B=$\{b_1, \ldots, b_k\}$ un tuple de k valeurs binaires. L'entropie conjointe de $\mathtt{x}$ est définie par :

$$\mathtt{H(x)} = -\sum_{B \in \{0,1\}^k} p(\mathtt{x} = B) \log p(\mathtt{x} = B)$$
où $p(\mathtt{x} = B)$ est la probabité jointe de $(\mathtt{x} = B)$

Soit $\mathtt{x}$ un k-itemset, $\mathtt{x}$ est un maximally informative k-itemset (*miki*) si et seulement si aucun k-itemset $\mathtt{y}$ a une plus grande entropie conjointe ($\mathtt{H(y)} > \mathtt{H(x)}$). Un miki est un motif de la taille spécifiée qui maximise l'entropie conjointe :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \mathtt{taille(x)} = k \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{H(y)} > \mathtt{H(x)}\}$$

8.1.2n Optimal Risk Patterns [Li *et al.*, 2005]

Le **risque relatif** d'un motif $\mathtt{x}$ est défini par $\mathtt{RR(x)} = \frac{\mathtt{freq_j(x)} \times (|\mathcal{T}| - \mathtt{freq(x)})}{(|\mathcal{T}_j| - \mathtt{freq_j(x)}) \times \mathtt{freq(x)}}$. Soit ψ_{risque} un seuil, un motif fréquent $\mathtt{x}$ est un *risk pattern* si et seulement si $\mathtt{RR(x)} \geq \psi_{risque}$, et $\mathtt{x}$ est un *optimal risk pattern* s'il n'existe aucune généralisation $\mathtt{y} \subset \mathtt{x}$ qui ait un **risque relatif** plus grand :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \mathtt{freq(x)} \geq minfr \wedge \mathtt{RR(x)} \geq \psi_{risque} \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{RR(y)} \geq \mathtt{RR(x)}\}$$

8.1.2o Les tuiles maximales [Geerts *et al.*, 2004]

Une tuile est maximale si et seulement si elle a une aire maximale, autrement dit il n'existe pas d'autres tuiles ayant une aire plus grande :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{freq}(\mathtt{x}) \geq 1 \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{aire}(\mathtt{y}) > \mathtt{aire}(\mathtt{x})\}$$

8.1.2p Strong Emerging Patterns [Soulet *et al.*, 2004]

Soit ρ un seuil d'émergence, un motif émergent $\mathtt{x}$ est un *Strong Emerging Pattern (SEP)* en $\mathcal{T}_1$ si et seulement si aucune spécialisation $\mathtt{y} \supset \mathtt{x}$ tel que $\mathtt{freq_j}(\mathtt{y}) = \mathtt{freq_j}(\mathtt{x})$ a une $\mathtt{émergence_j}$ plus élevée :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{émergence_j}(\mathtt{x}) \geq \rho \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : y \supset \mathtt{x} \wedge \mathtt{freq_j}(\mathtt{y}) = \mathtt{freq_j}(\mathtt{x}) \wedge \mathtt{émergence_j}(\mathtt{y}) \geq \mathtt{émergence_j}(\mathtt{x})\}$$

8.1.2q Maximal Emerging Patterns [Wang *et al.*, 2004]

Soient $minfr$ un seuil de fréquence, et ρ un seuil de $\mathtt{émergence_j}$. Un motif fréquent émergent $\mathtt{x}$ est *Maximal Emerging Pattern (MaxEP)* si et seulement si il n'existe aucune spécialisation $\mathtt{y} \supset \mathtt{x}$ qui soit aussi fréquente et émergente :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{émergent_j}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \supset \mathtt{x} \wedge \mathtt{émergent_j}(\mathtt{y})\}$$

$$\text{où } \mathtt{émergent_j}(\mathtt{x}) \equiv (\mathtt{émergence_j}(\mathtt{x}) \geq \rho \wedge \mathtt{freq}(\mathtt{x}) \geq minfr)$$

8.1.2r Essential Jumping Emerging Patterns [Fan et Ramamohanarao, 2002]

Soit $minfr$ un seuil de fréquence, un motif fréquent $\mathtt{x}$ est un *Essential Jumping Emerging Pattern (EJEP)* si et seulement si $\mathtt{x}$ est un JEP et aucune généralisation $\mathtt{y} \subset \mathtt{x}$ fréquente est aussi un JEP :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{jep_j}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{jep_j}(\mathtt{y})\}$$

$$\text{où } \mathtt{jep_j}(\mathtt{x}) \equiv (\mathtt{freq}(\mathtt{x}) = \mathtt{freq_j}(\mathtt{x}) \wedge \mathtt{freq}(\mathtt{x}) \geq minfr)$$

8.1.2s Règles de caractérisation [Crémilleux et Boulicaut, 2002]

Soient $minfr$ un seuil de fréquence et δ un entier. Une règle de caractérisation $\mathtt{x} \rightarrow \mathtt{j}$ est une règle fréquente où la prémisse $\mathtt{x}$ est minimale pour caractériser la classe $\mathtt{j}$ en acceptant un nombre d'exceptions contrôlé par δ. Plus formellement, cela signifie qu'il n'existe pas une règle $\mathtt{y} \rightarrow \mathtt{j}$ avec $\mathtt{y} \subset \mathtt{x}$ et $\mathtt{freq}(\mathtt{y}) - \mathtt{freq_j}(\mathtt{y}) \leq \delta$. Autrement dit :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \ \mathtt{caractérisation_j}(\mathtt{x}) \wedge \nexists\, \mathtt{y} \in \mathcal{L}_{\mathcal{I}} : \mathtt{y} \subset \mathtt{x} \wedge \mathtt{caractérisation_j}(\mathtt{y})\}$$

$$\text{où } \mathtt{caractérisation_j}(\mathtt{x}) \equiv (\mathtt{freq_j}(\mathtt{x}) \geq minfr \wedge (\mathtt{freq}(\mathtt{x}) - \mathtt{freq_j}(\mathtt{x}) \leq \delta))$$

Lorsque $\delta = 0$, on notera qu'une règle de caractérisation correspond à un Essential Jumping Emerging Pattern.

8.2 Une approche générique pour extraire les MO

Cette section montre comment l'extraction des MO peut être modélisée et résolue à l'aide des CSP dynamiques. À chaque étape, dès qu'une solution est trouvée, une nouvelle contrainte est ajoutée dynamiquement afin de rechercher une nouvelle solution qui soit de meilleure qualité (au sens de $\rhd$). Le processus s'arrête lorsqu'aucune meilleure solution ne peut plus être obtenue. Notre approche est générique et peut donc s'appliquer à l'extraction de n'importe quelle sorte de MO.

La section 8.2.1 montre comment un MO peut être modélisé et résolu à l'aide d'un CSP dynamique. La section 8.2.2 explicite la mise en œuvre sur les différentes sortes de MO présentées à la section 8.1.2.

8.2.1 Extraction des MO à l'aide des CSP dynamiques

Considérons la séquence $P_1, P_2, \ldots, P_n$ de CSP où chaque $P_i = (\{\mathtt{x}\}, \mathcal{L}_\mathcal{I}, \mathtt{q}_i(\mathtt{x}))$ et
— $\mathtt{q}_1(\mathtt{x}) = \mathtt{élémentaire}(\mathtt{x})$,
— $\mathtt{q}_{i+1}(\mathtt{x}) = \mathtt{q}_i(\mathtt{x}) \wedge \phi(s_i, \mathtt{x})$ où s_i est la première solution à la requête $\mathtt{q}_i(\mathtt{x})$.
Les contraintes $\phi(s_i, \mathtt{x})$ pour $i \in [1..n]$ imposent successivement que tous les motifs s_i obtenus ne soient pas meilleurs (au sens de $\rhd$) que le motif recherché $\mathtt{x}$ (si un s_i était meilleur que $\mathtt{x}$, alors $\mathtt{x}$ ne pourrait pas être un MO). Ainsi, à l'étape $(i+1)$ on ajoutera la contrainte $\phi(s_i, \mathtt{x})$ définie par :

$$\phi(s_i, \mathtt{x}) = (s_i \not\rhd \mathtt{x})$$

En conséquence, aucun des motifs s_1, s_2, …, s_i obtenus jusqu'à l'étape i ne peut être meilleur que $\mathtt{x}$ (preuve immédiate par induction). Les nouvelles contraintes $\phi(s_i, \mathtt{x})$ ajoutées dynamiquement permettent de réduire l'espace de recherche. L'extraction s'arrête lorsqu'aucun meilleur motif ne peut être obtenu, i.e. il existe n tel que $\mathtt{q}_{n+1}(\mathtt{x})$ n'a pas de solution.

Cependant, de manière analogue à l'extraction des skypatterns (cf. la section 6.3.1) et des skypatterns souples (cf. la section 6.3.3), tous les motifs extraits s_1, s_2, …, s_n ne sont pas nécessairement optimaux selon $\rhd$. Certains peuvent être des motifs *intermédiaires*, i.e. utiles uniquement pour améliorer l'élagage de l'espace de recherche. Ces derniers (i.e. les motifs s_i pour lesquels il existe s_j $(1 \leq i < j \leq n)$ tel que $s_j \rhd s_i$) sont filtrés par une étape de post-traitement. Ainsi, l'extraction s'effectue en deux étapes :

1. Calculer l'ensemble $\{s_1, s_2, \ldots, s_n\}$ des candidats à l'aide d'un CSP dynamique,
2. Supprimer tous les candidats s_i qui ne sont pas des MO.

8.2.2 Exemples

Cette section présente, pour chaque exemple de MO introduit à la section 8.1.2, la contrainte de base $\mathtt{élémentaire}(\mathtt{x})$ ainsi que l'ensemble de contraintes $\phi(s_i, \mathtt{x})$ ajoutées dynamiquement, qui lui sont associées.

8.2.2a Les motifs fermés

Soient $minfr$ un seuil de fréquence et $\mathtt{x}$ le motif inconnu. Alors, les contraintes associées sont :

$$\begin{aligned} \mathtt{élémentaire}(\mathtt{x}) &= (\mathtt{freq}(\mathtt{x}) \geq minfr) \\ \\ \phi(s_i, \mathtt{x}) &= (s_i \not\rhd \mathtt{x}) \\ &= \neg(s_i \supset \mathtt{x} \wedge \mathtt{freq}(s_i) = \mathtt{freq}(\mathtt{x})) \\ \phi(s_i, \mathtt{x}) &= (s_i \not\supset \mathtt{x}) \vee (\mathtt{freq}(s_i) \neq \mathtt{freq}(\mathtt{x})) \end{aligned}$$

Sur cet exemple, les fermés sont définis suivant la mesure de fréquence. Pour rechercher les fermés pour une mesure m quelconque, il suffit de remplacer `freq` par `m`.

8.2.2b Les skypatterns

Soient M un ensemble de mesures et $\mathtt{x}$ le motif inconnu. Alors, les contraintes associées sont :

$$\begin{aligned}
\texttt{élémentaire}(\mathtt{x}) &= \texttt{fermé}_{M'}(\mathtt{x}) \\
\\
\phi(s_i, \mathtt{x}) &= (s_i \not\trianglerighteq \mathtt{x}) \\
&= (s_i \not\succ_M \mathtt{x}) \\
\phi(s_i, \mathtt{x}) &= \left(\bigvee_{m \in M} \mathtt{m}(s_i) < \mathtt{m}(\mathtt{x})\right) \vee \left(\bigwedge_{m \in M} \mathtt{m}(s_i) = \mathtt{m}(\mathtt{x})\right) \quad \text{(cf. la section 6.3.1)}
\end{aligned}$$

8.2.2c Les skypatterns souples

Soient M un ensemble de mesures et $\mathtt{x}$ le motif inconnu.

Edge-skypatterns Alors, les contraintes associées sont :

$$\begin{aligned}
\texttt{élémentaire}(\mathtt{x}) &= \texttt{fermé}_{M'}(\mathtt{x}) \\
\\
\phi(s_i, \mathtt{x}) &= (s_i \not\trianglerighteq \mathtt{x}) \\
&= (s_i \not\gg_M \mathtt{x}) \\
\phi(s_i, \mathtt{x}) &= \left(\bigvee_{m \in M} m(s_i) \leq m(\mathtt{x})\right) \quad \text{(cf. la section 6.3.3)}
\end{aligned}$$

δ-skypatterns Alors, les contraintes associées sont :

$$\begin{aligned}
\texttt{élémentaire}(\mathtt{x}) &= \texttt{fermé}_{M'}(\mathtt{x}) \\
\\
\phi(s_i, \mathtt{x}) &= (s_i \not\trianglerighteq \mathtt{x}) \\
&= (s_i \not\succ^{\delta}_M \mathtt{x}) \\
\phi(s_i, \mathtt{x}) &= \left(\bigvee_{m \in M} (1-\delta) \times m(s_i) \leq m(\mathtt{x})\right) \quad \text{(cf. la section 6.3.3)}
\end{aligned}$$

La contrainte $\texttt{fermé}_M(\mathtt{x})$ impose que $\mathtt{x}$ soit un motif fermé pour M (cf. la section 6.3.4).

8.2.2d Les top-k

Soit m une mesure d'intérêt, l'ensemble des top-k (cf. la section 8.1.2i) est défini par :

$$\{\mathtt{x} \in \mathcal{L}_{\mathcal{I}} \mid \texttt{freq}(\mathtt{x}) \geq 1 \wedge \nexists\, \mathtt{y}_1, \ldots, \mathtt{y}_k \in \mathcal{L}_{\mathcal{I}} : \underbrace{\forall 1 \leq i \leq k, \mathtt{m}(\mathtt{y}_i) > \mathtt{m}(\mathtt{x})}_{\min_{1 \leq i \leq k} \{\mathtt{m}(\mathtt{y}_i)\} > \mathtt{m}(\mathtt{x})}\}$$

D'où $\texttt{élémentaire}(\mathtt{x}) = \texttt{freq}(\mathtt{x}) \geq 1$ et pour chaque solution trouvée s_i est stockée dans une liste S. Au départ, le premier motif $\mathtt{x}$ tel que $\texttt{freq}(\mathtt{x}) \geq 1$ est recherché. Tant que $i < k$, le nombre de motifs

MO	$\texttt{élémentaire}(\texttt{x})$	$\phi(s_i, \texttt{x})$
Motifs fermés	$\texttt{freq}(\texttt{x}) \geq minfr$	$s_i \not\supset \texttt{x} \vee \texttt{freq}(s_i) \neq \texttt{freq}(\texttt{x})$
Motifs libres	$\texttt{freq}(\texttt{x}) \geq minfr$	$s_i \not\subset \texttt{x} \vee \texttt{freq}(s_i) \neq \texttt{freq}(\texttt{x})$
Motifs maximaux fréquents	$\texttt{freq}(\texttt{x}) \geq minfr$	$s_i \not\supset \texttt{x}$
Motifs minimaux non-fréquents	$\texttt{freq}(\texttt{x}) < maxfr$	$s_i \not\subset \texttt{x}$
Skypatterns	$\texttt{fermé}_{M'}(\texttt{x})$	$\left(\bigvee_{m \in M} \texttt{m}(s_i) < \texttt{m}(\texttt{x})\right) \vee \left(\bigwedge_{m \in M} \texttt{m}(s_i) = \texttt{m}(\texttt{x})\right)$
Edge-skypatterns	$\texttt{fermé}_{M'}(\texttt{x})$	$\left(\bigvee_{m \in M} m(s_i) \leq m(\texttt{x})\right)$
δ-skypatterns	$\texttt{fermé}_{M'}(\texttt{x})$	$\left(\bigvee_{m \in M} (1-\delta) \times m(s_i) \leq m(\texttt{x})\right)$
Motifs pics	$\texttt{freq}(\texttt{x}) \geq 1$	$d(\texttt{x}, s_i) > \delta \vee \varepsilon \times \texttt{m}(s_i) \leq \texttt{m}(\texttt{x})$
top-k	$\texttt{freq}(\texttt{x}) \geq 1$	$\texttt{m}(\texttt{x}) \geq \min_{s' \in S} \texttt{m}(s')$ si $i \geq k$ $vrai$ sinon
The N-most interesting k-itemsets	$\texttt{taille}(\texttt{x}) = k$	$\texttt{freq}(\texttt{x}) \geq \min_{s' \in S} \texttt{freq}(s')$ si $i \geq N$ $vrai$ sinon
Découverte de sous-groupes pertinents	$\texttt{freq}(\texttt{x}) \geq minfr$	$T_1(s_i) \supseteq T_1(\texttt{x}) \wedge$ $T_2(s_i) \subseteq T_2(\texttt{x}) \wedge$ $(T(s_i) = T(\texttt{x}) \implies s_i \subset \texttt{x})$
Pattern compression problem	$\texttt{fermé}(\texttt{x})$	$d(\texttt{x}, s_i) > \delta \vee s_i \subset \texttt{x}$
Maximally informative k-itemset	$\texttt{taille}(\texttt{x}) = k$	$\texttt{H}(s_i) \leq \texttt{H}(\texttt{x})$
Optimal Risk Patterns	$\texttt{freq}(\texttt{x}) \geq minfr \wedge \texttt{RR}(\texttt{x}) \geq \psi_{risque}$	$s_i \not\subset \texttt{x} \vee \texttt{RR}(s_i) < \texttt{RR}(\texttt{x})$
Les tuiles maximales	$\texttt{freq}(\texttt{x}) \geq 1$	$\texttt{aire}(s_i) \leq \texttt{aire}(\texttt{x})$
Strong Emerging Patterns	$\texttt{émergence}_{\texttt{j}}(\texttt{x}) \geq \rho$	$s_i \not\supset \texttt{x} \vee \texttt{freq}_{\texttt{j}}(s_i) \neq \texttt{freq}_{\texttt{j}}(\texttt{x}) \vee \texttt{émergence}_{\texttt{j}}(s_i) < \texttt{émergence}_{\texttt{j}}(\texttt{x})$
Maximal Emerging Patterns	$\texttt{émergence}_{\texttt{j}}(\texttt{x}) \geq \rho \wedge \texttt{freq}(\texttt{x}) \geq minfr$	$s_i \not\supset \texttt{x}$
Essential Jumping Emerging Patterns	$\texttt{freq}(\texttt{x}) = \texttt{freq}_{\texttt{j}}(\texttt{x}) \wedge \texttt{freq}(\texttt{x}) \geq minfr$	$s_i \not\subset \texttt{x}$
Règles de caractérisation	$\texttt{freq}_{\texttt{j}}(\texttt{x}) \geq minfr \wedge (\texttt{freq}(\texttt{x}) - \texttt{freq}_{\texttt{j}}(\texttt{x}) \leq \delta)$	$s_i \not\subset \texttt{x}$

TABLE 8.1 – Contraintes élémentaires et contraintes ajoutées dynamiquement pour chaque MO.

recherchés n'est pas encore atteint. Il est alors trop tôt pour contraindre la recherche et en conséquence $\phi(s_i, \mathtt{x}) = vrai$. Dès que la k-ème solution est obtenue, il faut imposer que le nouveau motif $\mathtt{x}$ recherché soit meilleur (selon m) qu'au moins un des k meilleurs motifs déjà obtenus en postant la contrainte $\phi(s_i, \mathtt{x})$ et en retirant la solution avec la plus petite valeur selon m i.e. $S \leftarrow S \setminus \{\arg\min_{s' \in S} \mathtt{m}(s')\}$.

$\phi(s_i, \mathtt{x})$ est alors définie par :

$$\phi(s_i, \mathtt{x}) = \begin{cases} \mathtt{m}(\mathtt{x}) \geq \min_{s' \in S} \mathtt{m}(s') & si \quad i \geq k \\ vrai & sinon \end{cases}$$

8.2.2e Essential Jumping Emerging Patterns

Soient $minfr$ un seuil de fréquence et $\mathtt{x}$ le motif inconnu. Alors, les contraintes associées sont :

$$\begin{array}{rcl} \mathtt{élémentaire}(\mathtt{x}) & = & \mathtt{jep_j}(\mathtt{x}) = (\mathtt{freq}(\mathtt{x}) = \mathtt{freq_j}(\mathtt{x}) \wedge \mathtt{freq}(\mathtt{x}) \geq minfr) \\ \\ \phi(s_i, \mathtt{x}) & = & (s_i \not\rhd \mathtt{x}) \\ & = & \neg(s_i \subset \mathtt{x} \wedge \mathtt{jep_j}(s_i)) \quad (s_i \models \mathtt{q_1}(\mathtt{x}) \implies \mathtt{jep_j}(s_i) \equiv \top) \\ \phi(s_i, \mathtt{x}) & = & (s_i \not\subset \mathtt{x}) \end{array}$$

8.2.2f Autres exemples

Tous les MO présentés à la section 8.1.2 peuvent être extraits par notre approche. Le tableau 8.1 résume, pour chaque MO, les contraintes $\mathtt{élémentaire}(\mathtt{x})$ et $\phi(s_i, \mathtt{x})$ qui lui sont associées. Pour un MO défini par la préférence $\rhd$, les contraintes $\phi(s_i, \mathtt{x})$ traduisent la relation $(s_i \not\rhd \mathtt{x})$. Ainsi, les CSP dynamiques permettent de concevoir une approche générique capable d'extraire toutes sortes de MO.

8.3 Travaux relatifs

Dans [Crémilleux et Soulet, 2008], l'idée d'une approche générique (appelée *approximer-et-pousser*) pour traiter des problèmes d'extraction de motifs a été introduite, l'implémentation repose sur une contrainte locale qui est poussée mais qu'il faut définir. Le principe de cette approche a été discutée à la section 1.5.2 (cf. page 31).

Dans [Négrevergne *et al.*, 2013], une algèbre fondée sur la notion de dominance binaire a été définie pour modéliser la relation de préférence entre deux motifs. Contrairement à notre relation de préférence (cf. la définition 1.24), qui est un ordre strict (i.e. une relation irréflexive et transitive) et partiel (i.e. ayant des éléments incomparables), la dominance binaire est un pré-ordre (i.e. une relation réflexive et transitive). Bien que cette différence puisse paraître subtile, dans [Négrevergne *et al.*, 2013], les auteurs doivent trouver le pré-ordre associé à la relation de préférence induite par la définition de chaque problème, afin de reformuler le problème dans leur algèbre.

Le tableau 8.2 compare les deux relations de préférence (préférence basée sur la définition 1.24 vs. préférence basée sur la dominance binaire) sur quelques exemples :

Bien qu'il existe toujours un pré-ordre associé à un ordre strict [Caspard *et al.*, 2012] et vice-versa, toutefois, pour certains problèmes, trouver un tel pré-ordre peut s'avérer une tâche très difficile. Le

	Préférence (ordre strict et partiel) $\mathtt{y} \rhd \mathtt{x}$	Dominance [Négrevergne *et al.*, 2013] (pré-ordre) y domine x
Motifs maximaux	$\mathtt{y} \supset \mathtt{x}$	$\mathtt{y} \supseteq \mathtt{x}$
Motifs fermés	$\mathtt{y} \supset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})$	$\mathtt{y} \supseteq \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})$
Motifs libres	$\mathtt{y} \subset \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})$	$\mathtt{y} \subseteq \mathtt{x} \wedge \mathtt{freq}(\mathtt{y}) = \mathtt{freq}(\mathtt{x})$
Skypatterns	$\mathtt{y} \succ_M \mathtt{x}$ $\equiv \left\{ \begin{array}{ll} \forall m \in M, m(\mathtt{y}) \geq m(\mathtt{x}) & \wedge \\ \exists m' \in M, m'(\mathtt{y}) > m'(\mathtt{x}) & \end{array} \right.$	$\equiv \forall m \in M, m(\mathtt{y}) \geq m(\mathtt{x})$

TABLE 8.2 – Préférence basée sur la définition 1.24 vs. préférence basée sur la dominance binaire.

		Préférence (ordre strict et partiel) $\mathtt{y} \rhd \mathtt{x}$	Dominance [Négrevergne *et al.*, 2013] (pré-ordre) y domine x
Skypatterns souples	Edge-Skypatterns	$\mathtt{y} \gg_M \mathtt{x}$ $\equiv \forall m \in M, m(\mathtt{y}) > m(\mathtt{x})$	?
	δ-Skypatterns	$\mathtt{y} \succ^{\delta}_M \mathtt{x}$ $\equiv \forall m \in M, (1-\delta)m(\mathtt{y}) > m(\mathtt{x})$	?
Pattern Compression Problem		$d(\mathtt{x}, \mathtt{y}) \leq \delta \wedge \mathtt{y} \not\subset \mathtt{x}$	?

TABLE 8.3 – Problèmes où le pré-ordre n'est pas simple à trouver.

tableau 8.3 montre quelques exemples de problèmes où il n'est pas facile de trouver le pré-ordre associé à la préférence exprimée par le problème.

Par ailleurs, tous les MO que nous avons listés (cf. le tableau 8.1) ont été définis dans la littérature en utilisant des préférences avec des ordres stricts partiels (même si ces problèmes n'ont pas été explicitement définis ainsi). Il nous semble donc naturel d'utiliser une relation de préférence basée sur un ordre strict partiel. De plus, sur certains problèmes, comme les skypatterns, les auteurs [Négrevergne *et al.*, 2013] utilisent une mise en oeuvre de la relation de dominance basée sur un ordre strict partiel pour dériver la contrainte $\phi(s_i, \mathtt{x})$ postée dynamiquement et cela afin de ne pas rater de solutions.

Enfin, un autre inconvénient de cette algèbre est qu'elle ne peut pas être appliquée aux cas où n motifs sont comparés (e.g. *top-k*, N-most).

8.4 Expérimentations

Nous montrons la faisabilité de notre approche à partir de trois sortes de MO : les fermés/libres/-maximaux, les skypatterns et les motifs pics. Pour chacun d'eux, nous comparons les performances de notre approche générique avec une ou plusieurs méthodes ad hoc. Les jeux de donnés utilisés sont ceux de l'*UCI*. Toutes les expérimentations ont été réalisées sous Linux avec un processeur Core i3 à 2.13 GHz et une mémoire vive de 4 Go. PPC+MO a été développé en *Gecode* (cf. l'annexe A.1).

8.4.1 Motifs fermés, libres et maximaux.

Nous comparons notre approche PPC+MO avec les *meilleurs* extracteurs : ECLAT [Borgelt, 2003], et LCM [Uno *et al.*, 2004] (uniquement pour les motifs fermés). Les expérimentations ont été menées en faisant varier le seuil de fréquence $minfr$. La figure 8.1 montre que les méthodes ad hoc pour ces motifs (ECLAT et LCM) sont plus performantes que notre approche générique. Ce résultat n'est pas surprenant

et s'explique car les chercheurs ont fourni d'efforts depuis 15 ans sur l'extraction de représentations condensées, malgré tout il montre la faisabilité de notre approche.

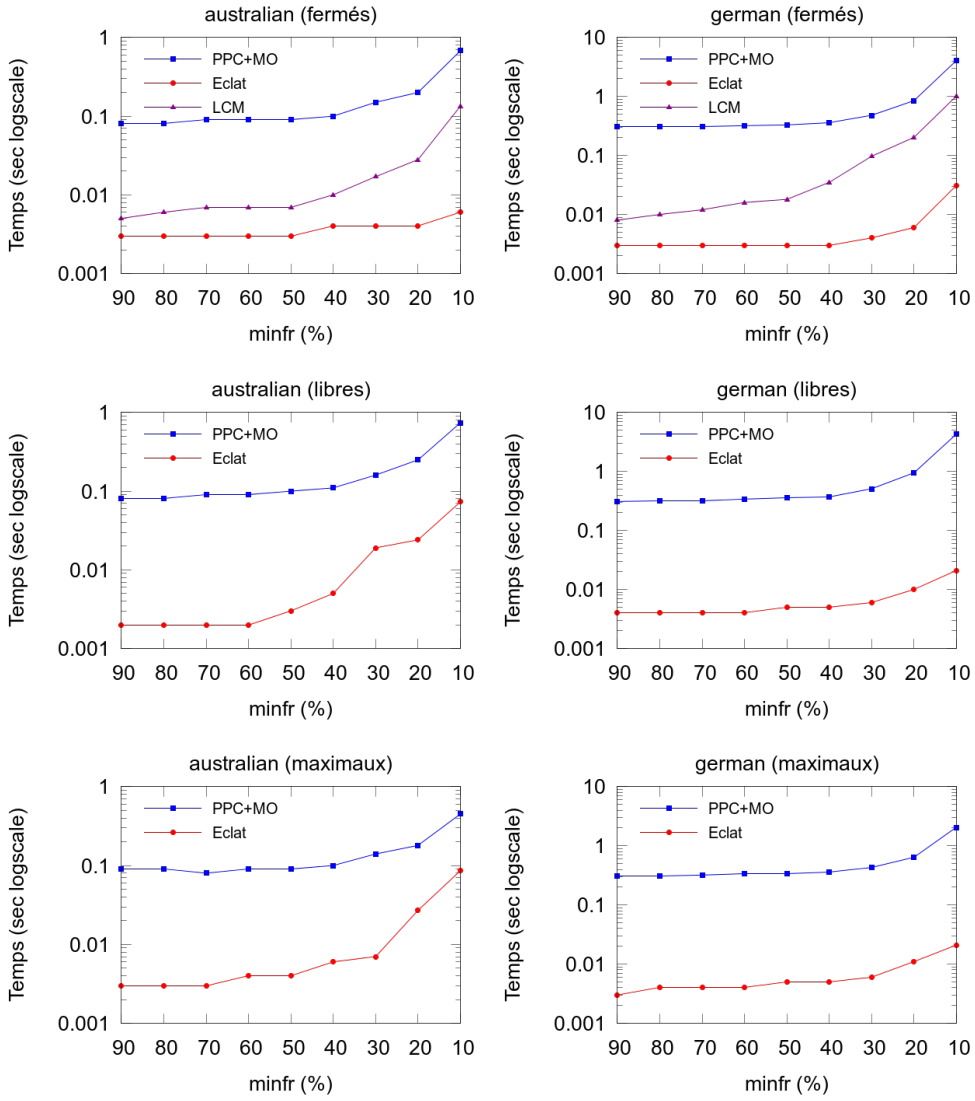


FIGURE 8.1 – Comparaison des temps de calcul (motifs fermés, libres et maximaux).

8.4.2 Skypatterns

Comme le montrent les expérimentations menées à la section 6.4.1 pour les jeux de données de l'*UCI*, et à la section 6.5.1 pour notre étude de cas consacrée aux toxicophores, notre approche générique et `AETHERIS` ont des performances similaires.

8.4.3 Motifs pics

Nous avons comparé PPC+MO avec l'approche développée en *QeCode* dans [Khiari *et al.*, 2012], qui est la seule méthode connue pour extraire les motifs pics (cf. l'annexe A.2). Les expérimentations ont été menées en faisant varier le seuil de fréquence $minfr$, pour différentes valeurs de ε. La figure 8.2 montre que PPC+MO surpasse nettement *QeCode* : l'évolution des temps de calcul semble quasi-linéaire pour PPC+MO alors qu'elle est visiblement exponentielle pour *QeCode*.

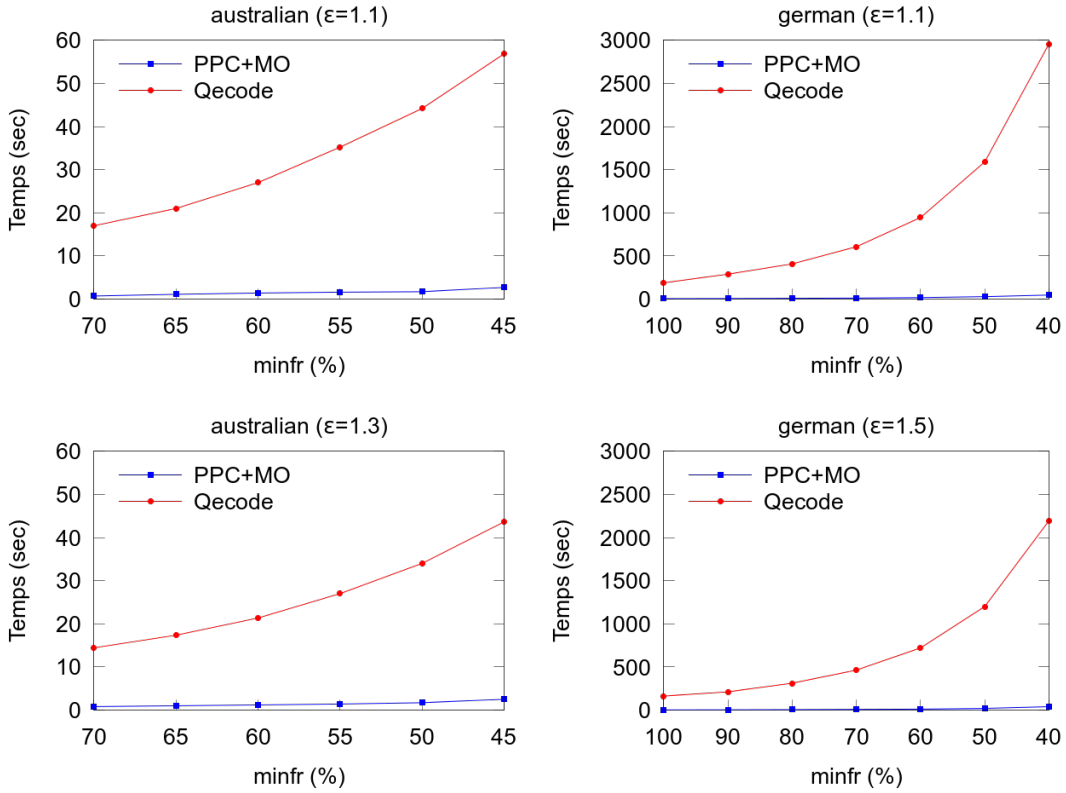


FIGURE 8.2 – Comparaison des temps de calcul (motifs pics).

8.4.4 Synthèse

Notre méthode générique est moins performante pour les motifs "simples" dont l'extraction a été profondément étudiée (e.g. fermés). En revanche, pour l'extraction d'ensembles de motifs plus élaborés comme les skypatterns, notre approche s'avère aussi compétitive voire plus performante que les méthodes ad hoc existantes (cf. la section 8.4.3).

8.5 Conclusion

Dans ce chapitre, nous avons introduit la notion de MO et nous avons montré que les MO permettaient de modéliser de nombreux problèmes d'extraction de motifs : skypatterns, *top-k*, motifs fermés, ... La résolution à l'aide des CSP dynamiques rend notre approche générique. Nous avons comparé notre

approche avec des méthodes ad hoc et montré que notre approche générique est concurrentielle. La déclarativité de notre approche fondée sur l'utilisation des CSP dynamiques ouvre la voie à la définition et la découverte de nouveaux ensembles de motifs.

Troisième partie

Conclusion et perspectives

"La prédiction est très difficile, en particulier à propos de l'avenir."

Niels Bohr

Conclusions

Les travaux présentés dans ce mémoire ont pour objectif de s'attaquer aux deux constats suivants concernant l'extraction de motifs en fouille de données :

i) la rigidité du cadre actuel, qui fait qu'une solution potentiellement intéressante n'est pas prise en compte dès qu'une contrainte et/ou une valeur seuil pour une contrainte sur une mesure n'est pas vérifiée.

ii) la difficulté, pour l'utilisateur, d'exprimer des préférences, afin de pouvoir prendre en compte des priorités ou encore des incertitudes.

Grâce aux travaux existants en PPC sur la relaxation de contraintes et les préférences entre solutions, nous avons pu apporter quatre principales contributions que nous rappelons dans les sections suivantes.

1 Extraction de motifs sous contraintes souples de seuil

Le chapitre 5 a introduit la notion de contrainte souple de seuil en définissant plusieurs sémantiques de violation. Puis, nous avons montré comment de telles contraintes peuvent êtres mises en œuvre en utilisant le modèle PPC de la relaxation disjonctive. Nous avons étendu cette approche à l'extraction des top-k motifs souples. Différentes expérimentations ont été menées pour la découverte de fragments moléculaires toxicophores.

Ce chapitre met en évidence les apports de la souplesse dans le processus d'extraction de motifs. Il montre comment la souplesse permet de découvrir des motifs intéressants qui ne seraient pas extraits autrement. Notre approche procure aussi un moyen naturel de combiner, dans un même cadre, les mesures usuelles en extraction de motifs, avec des mesures provenant de la connaissance du domaine applicatif. La pertinence et l'efficacité de notre approche sont mises en évidence par une étude de cas en chemoinformatique concernant la découverte de fragments moléculaires toxicophores. Les résultats expérimentaux démontrent l'avantage d'utiliser des contraintes souples de seuil pour la découverte de toxicophores.

2 Extraction de skypatterns souples

Le chapitre 6 a introduit la notion de skypattern souple, en distinguant les edge-skypatterns qui sont à la bordure de la zone de dominance, et les δ-skypatterns qui sont à une distance δ de cette bordure. Puis, nous avons proposé une méthode générique et efficace, reposant sur les CSP Dynamiques, qui permet d'extraire aussi bien les skypatterns (durs) que les skypatterns souples. Les expérimentations menées montrent l'apport des skypatterns souples pour la découverte de toxicophores.

De manière analogue au chapitre 5, ce chapitre met en évidence les apports de la souplesse dans le processus d'extraction de motifs. Tout d'abord, les skypatterns ne nécessitent pas de seuil pour les

contraintes portant sur les mesures, et la notion de dominance offre un intérêt global avec une sémantique facilement compréhensible par l'utilisateur. De plus, la souplesse permet de découvrir des motifs intéressants qui ne seraient pas extraits autrement.

3 Extraction de cubes de skypatterns

Le chapitre 7 a introduit la notion de cube de skypatterns et propose deux méthodes pour construire un tel cube : tout d'abord, une méthode bottom-up fondée sur deux règles de dérivation, les skypatterns non-dérivables étant calculés à la volée ; puis, une seconde méthode utilisant les skypatterns souples, le problème de construction du cube de skypatterns pouvant être facilement ramené au calcul d'une cube de skylines. Ces deux méthodes sont validées et comparées expérimentalement sur des jeux de données de l'*UCI* et au travers d'une étude de cas consacrée à la découverte de composants mutagènes.

Ici, l'apport de la souplesse se limite à la seconde méthode de calcul du cube de skypatterns (par relaxation/approximation), celle-ci étant plus efficace que la méthode bottom-up, et permettant donc de traiter des cubes de plus grande taille.

4 Modélisation et extraction de motifs optimaux

Le chapitre 8 a introduit la notion de motifs optimaux (MO) qui sont les meilleurs motifs selon une préférence définie par l'utilisateur. Tout d'abord, nous montrons que les MO permettent d'exprimer de nombreux problèmes d'extraction d'ensembles de motifs. Puis, nous proposons une méthode générique, reposant sur les CSP dynamiques, permettant d'extraire n'importe quelle sorte de MO, en montrant que chaque sorte de MO peut être caractérisée par une contrainte initiale et un ensemble de contraintes ajoutées dynamiquement. Une étude expérimentale, menée sur plusieurs sortes de MO et différents jeux de données de l'*UCI*, illustre et discute la qualité des performances obtenues par rapport aux méthodes ad hoc.

Nous pensons que la déclarativité et la généricité de notre approche ouvrent la voie à la définition et à la découverte de nouveaux ensembles de motifs.

Perspectives

Les perspectives de recherche s'inscrivent dans la continuité des travaux menés, que ce soit pour introduire de la souplesse dans d'autres tâches en fouille de données, pour utiliser les skypatterns et les skypatterns souples pour les tâches de recommandation, ou pour naviguer dans le cube de skypatterns pour mener une étude fine des ensembles de mesures.

Relaxation de contraintes

Introduction de la souplesse dans d'autres tâches de fouille de données : Dans le cadre de cette thèse, nous nous sommes intéressés à l'introduction de la souplesse pour l'extraction de motifs ensemblistes. Toutefois, plusieurs travaux récents utilisent la PPC pour d'autres tâches, comme par exemple :

— la fouille de séquences [Coquery *et al.*, 2012, Jabbour *et al.*, 2013a, Kemmar *et al.*, 2014],

— le clustering [Dao *et al.*, 2013, Babaki *et al.*, 2014].

Nous souhaitons étudier les apports de la souplesse et leur mise en œuvre pour ces deux tâches. Par exemple, pour le clustering sous contraintes, il est parfois très difficile de satisfaire les contraintes de non-recouvrement ou les contraintes d'équilibrage.

Extension à d'autres types de contraintes : Jusqu'à présent, nous n'avons cherché à relaxer que les contraintes de seuil. Il serait intéressant de s'intéresser à d'autres types de contraintes. Par exemple, pour les contraintes ensemblistes, il est possible de définir des sémantiques de violation appropriées :

— Pour la contrainte d'égalité : la sémantique de violation peut être définie comme étant le cardinal de la différence symétrique entre deux motifs (ensemblistes) :

$$X_p = X_q \rightarrow z = |X_p \setminus X_q| + |X_q \setminus X_p|$$

— De manière analogue, pour la contrainte d'inclusion :

$$X_p \subseteq X_q \rightarrow z = |X_q \setminus X_p|$$

De nouveaux usages et de nouvelles formes de skypatterns

Skypatterns souples pour les tâches de recommandation : Les skypatterns souples peuvent être exploités pour des tâches de recommandation. En effet, en plus des skypatterns durs, il serait intéressant de recommander à l'utilisateur d'autres solutions "proches", comme les skypatterns souples, qui peuvent constituer des solutions de (très) bon compromis ou dont l'intérêt deviendra crucial si, de façon imprévue, des solutions représentées par des skypatterns durs ne sont plus disponibles.

top-k skypatterns et top-k skypatterns souples : Il existe plusieurs travaux portant sur l'extraction des top-k dans les requêtes skylines [Tao *et al.*, 2007, Yiu et Mamoulis, 2009], permettant d'extraire les k meilleurs tuples Pareto-optimaux selon une mesure d'intérêt. Nous pensons que de telles méthodes peuvent être conçues pour les skypatterns et aux skypatterns souples.

Autres types de dominance : Outre la Pareto-dominance, la dominance stricte et la δ-dominance, il serait intéressant d'étudier d'autres types de dominance, comme par exemple la k-dominance [Chan *et al.*, 2006], ou des dominances issues de l'optimisation multi-critères, comme l'ϵ-dominance [Xia *et al.*, 2008], ou l'α-dominance [Batista *et al.*, 2011] ou encore la volume dominance [Le et Silva, 2007, Ehrgott *et al.*, 2009].

Cube de skypatterns

Navigation dans le cube de skypatterns : La navigation dans le cube de skypatterns est aussi une perspective très prometteuse. Les classes d'équivalence définies par rapport aux mesures utilisées sont déjà une façon de mettre en évidence le rôle de certaines mesures puisque ces classes font ressortir les ensembles de mesures donnant les mêmes ensembles de skypatterns.

Une méthode bottom-up pour le calcul du cube de skylines : Notre méthode bottom-up de calcul d'un cube de skypatterns, présentée au chapitre 7, nous semble pouvoir être facilement transposée au calcul d'un cube de skylines, tout en gardant les mêmes bonnes propriétés. Toutefois, il reste à évaluer l'efficacité d'une telle approche par rapport aux méthodes existantes. En effet, le calcul à la volée des skylines non-dérivables à l'aide des CSP dynamiques pourrait s'avérer pénalisant.

Modélisation et extraction de motifs optimaux

Spécification des préférences : Afin de faciliter l'expression de préférences par l'utilisateur, des méthodes efficaces pourraient être développées à partir de techniques connues (e.g. l'acquisition de contraintes [Bessiere *et al.*, 2014], ...).

Appliquer notre approche à d'autres tâches : Les préférences, et leur mise en oeuvre par les CSP dynamiques, pourraient être étendues à d'autres tâches, comme par exemple, l'extraction des top-k tuiles dans les flux de données [Lam *et al.*, 2014]. Ce problème étant défini sur des flux de données (et non plus des données statiques comme celles traitées dans ce mémoire), il serait aussi intéressant d'étudier comment la PPC permet de gérer les modifications des contraintes dues au changement des données entre les différentes fenêtres d'observation.

Modélisation

Autre cadre PPC : Les méthodes d'extraction présentées dans les chapitres 6, 7 et 8 reposent sur la modélisation sous forme de CSP dynamiques. Il serait intéressant de proposer une autre modélisation, qui éviterait notamment la construction (inutile) de candidats intermédiaires.

Autre modèle d'implémentation : Actuellement, toutes les méthodes d'extraction se basent sur l'encodage booléen proposé par [De Raedt *et al.*, 2008]. Il nous semblerait intéressant de proposer une autre modélisation afin d'améliorer l'efficacité temporelle et diminuer l'espace mémoire nécessaire, notamment pour tout ce qui concerne les contraintes réifiées.

Annexes

Annexe A

Outils PPC

A.1 *Gecode*

Gecode (GEneric COnstraint Development Environment) [42] est un environnement ouvert, libre et portable, pour le développement de systèmes et d'applications à base de contraintes.

Gecode [Gecode Team, 2006] est :

— **Ouvert :** *Gecode* est radicalement ouvert pour la programmation : il peut facilement être relié à d'autres systèmes. Il prend en charge la programmation de nouveaux propagateurs (comme la mise en œuvre de contraintes), des stratégies de branching et des technique de recherche. Nouvelles variables peuvent être programmées au même niveau d'efficacité avec des variables entières, variables ensemblistes, et les variables flottantes qui sont fournis avec *Gecode*.

— **Libre :** *Gecode* est distribué sous la licence MIT [43] et est répertorié comme logiciel libre par la FSF [44]. Toutes ses parties - y compris la documentation de référence, les implémentations de contraintes globales, et des exemples - sont disponibles en code source pour le téléchargement.

— **Portable :** *Gecode* est implémenté en C++ qui suit rigoureusement le standard C++. Il peut être compilé avec les compilateurs C++ modernes et fonctionne ainsi sur un large gamme de plates-formes.

Exemple A.2.
Nous illustrons la modélisation d'un CSP en *Gecode* par un exemple célèbre dans la communauté de la PPC. Il s'agit de résoudre le cryptogramme `SEND+MORE=MONEY`, où deux lettres différentes correspondent à deux chiffres distincts.
Ce cryptogramme est modélisé par le CSP suivant :

— $\mathcal{X} = \{S, E, N, D, M, O, R, Y\}$
— $\mathcal{D} = \{D_S = D_E = D_N = D_D = D_M = D_O = D_R = D_Y = \{0, 1, ..., 9\}\}$
—

$$\mathcal{C} = \begin{cases} \texttt{alldifferent}([S, E, N, D, M, O, R, Y]) & \wedge \\ S \neq 0 & \wedge \\ M \neq 0 & \wedge \\ (1000 \times S) + (100 \times E) + (10 \times N) + D + \\ (1000 \times M) + (100 \times O) + (10 \times R) + E = \\ (10000 \times M) + (1000 \times O) + (100 \times N) + (10 \times E) + Y \end{cases}$$

42. `http://www.gecode.org`
43. Massachusetts Institute of Technology
44. Free Software Foundation

La figure A.1 donne un modèle *Gecode* commenté du problème $SEND + MORE = MONEY$. Il s'agit d'une classe C++ et d'une fonction `main`. Dans la première, les variables et les contraintes sont déclarées et instanciées, et la résolution paramétrée. Dans la deuxième, on lance la méthode de résolution associée à la classe qu'on vient de définir.

```
1  class Money: public Script {
2  protected:
3     static const int nl = 8;     /// Number of letters
4     IntVarArray le;     /// Array of letters
5  public:
6     Money(const Options& opt): le(*this,nl,0,9) {
7      IntVar s(le[0]), e(le[1]), n(le[2]), d(le[3]), m(le[4]), o(le[5]), r(le[6]), y(le[7]);
8
9      rel(*this, s, IRT_NQ, 0);
10     rel(*this, m, IRT_NQ, 0);
11
12     distinct(*this, le, opt.icl());
13
14     rel(*this,             1000*s+100*e+10*n+d
15                  +         1000*m+100*o+10*r+e
16                 == 10000*m+1000*o+100*n+10*e+y,
17         opt.icl());
18
19     branch(*this, le, INT_VAR_SIZE_MIN, INT_VAL_MIN);
20   }
21
22   /// Constructor for cloning \a s
23   Money(bool share, Money& s): Script(share,s) {
24     le.update(*this, share, s.le);
25   }
26   /// Copy during cloning
27   virtual Space*
28   copy(bool share) {
29     return new Money(share,*this);
30   }
31 };
32
33 int main(int argc, char* argv[]) {
34   Options opt("SEND+MORE=MONEY");
35   opt.solutions(0);
36   opt.iterations(20000);
37   opt.parse(argc,argv);
38   Script::run<Money,DFS,Options>(opt);
39   return 0;
40 }
```

— l. 1-9 : déclaration d'une classe C++ dédiée au problème, instanciation du paramètre *nl* (nombre de lettres) et déclaration et instanciation des variables,
— l.11-12 : ajout des contraintes empêchant S et M de valoir 0,
— l. 14 : ajout d'une contrainte `alldifferent` portant sur l'ensemble des variables,
— l. 16-19 : ajout de la contrainte arithmétique spécifiant la relation de somme entre les variables,
— l. 21 : lancement de la recherche et définition des heuristiques,
— l. 25-32 : clonage de l'état courant de l'arbre de recherche,
— l. 35-42 : initialisation et lancement de la résolution.

FIGURE A.1 – Programme *Gecode* associé au cryptogramme `SEND+MORE=MONEY`.

A.2 *QeCode*

QeCode [Benedetti *et al.*, 2007, Benedetti *et al.*, 2008] [45] est un solveur open-source de Problèmes de Satisfaction de Contraintes Quantifiées (QCSP) implanté en *Gecode* (cf. la section précédente). Les QCSP sont une extension des CSP classiques, dans laquelle certaines variables peuvent être quantifiées universellement. Les QCSP permettent de modéliser l'incertitude sur les données ou la présence d'un adversaire. Les applications se trouvent dans la théorie de jeux, la manipulation de l'incertitude, la planification conforme, l'ordonnancement robuste, le model checking, ...

Exemple A.3. Motifs pic
Dans la section 1.4.3 (cf. page 27), nous avons présenté les motifs pics [Crémilleux et Soulet, 2008] comme exemple de requête définie à l'aide des contraintes globales.

Un motif pic est un motif qui possède une valeur, pour une mesure $\mathtt{m}$ donnée, assez grande par rapport à celles de tous ses voisins. Le voisinage d'un motif étant calculé par rapport à une distance d donnée. Un motif pic a ainsi un comportement exceptionnel par rapport à ses voisins.

Soit $\mathtt{m} : \mathcal{L}_\mathcal{I} \rightarrow \mathbb{R}$ une mesure, δ un entier et ε un réel, et soit d une distance, on a :

$$\mathtt{pic(x)} \equiv \begin{cases} vrai & si \ \ \forall \mathtt{y} \in \mathcal{L}_\mathcal{I} \ tel \ que \ d(\mathtt{x},\mathtt{y}) \leq \delta, \mathtt{m(x)} \geq \varepsilon \times \mathtt{m(y)} \\ faux & sinon \end{cases}$$

Pour modéliser ce problème, on introduit les variables suivantes :
— X_1 représente le motif $\mathtt{x}$ quantifiée existentiellement.
— X_2 représente le motif $\mathtt{y}$ représentant le voisinage de $\mathtt{x}$ et quantifiée universellement.
La requête $\mathtt{pic(x)}$ est alors modélisée par le QCSP suivant :

$$Q = \Big(\big[(\exists, X_1, C_1), \ (\forall, X_2, C_2) \big], \ Goal = C_3 \Big)$$
où : $C_1 = \emptyset$
$C_2 = d(X_1, X_2) \leq \delta$
$C_3 = \mathtt{m}(X_1) \geq \varepsilon \times \mathtt{m}(X_2)$

45. `http://www.univ-orleans.fr/lifo/software/qecode/QeCode.html`

Bibliographie

[Agrawal et Srikant, 1994] Agrawal, R. et Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *In* Bocca, J. B., Jarke, M. et Zaniolo, C., éditeurs : *VLDB*, pages 487–499. Morgan Kaufmann. 13, 17

[Antunes et Oliveira, 2004] Antunes, C. et Oliveira, A. L. (2004). Constraint relaxations for discovering unknown sequential patterns. *In* Goethals, B. et Siebes, A., éditeurs : *KDID*, volume 3377 de *Lecture Notes in Computer Science*, pages 11–32. Springer. 71

[Auer et Bajorath, 2006] Auer, J. et Bajorath, J. (2006). Emerging chemical patterns : A new methodology for molecular classification and compound selection. *Journal of Chemical Information and Modeling*, 46(6):2502–2514. 72

[Babaki *et al.*, 2014] Babaki, B., Guns, T. et Nijssen, S. (2014). Constrained clustering using column generation. *In Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*, pages 438–454. 153

[Bajorath, 2008] Bajorath, J. (2008). Simulation of sequential screening experiments using emerging chemical patterns. *Medicinal Chemistry*, 4:80–90. 72

[Bastide *et al.*, 2000] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G. et Lakhal, L. (2000). Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75. 137

[Batista *et al.*, 2011] Batista, L. S., Campelo, F., Guimarães, F. G. et Ramírez, J. A. (2011). A comparison of dominance criteria in many-objective optimization problems. *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, pages 2359–2366. IEEE. 154

[Bayardo, 1998] Bayardo, R. (1998). Efficiently mining long patterns from databases. *In SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 85–93. 137

[Benedetti *et al.*, 2007] Benedetti, M., Lallouet, A. et Vautard, J. (2007). QCSP made practical by virtue of restricted quantification. *In* Veloso, M. M., éditeur : *IJCAI*, pages 38–43. 161

[Benedetti *et al.*, 2008] Benedetti, M., Lallouet, A. et Vautard, J. (2008). Quantified constraint optimization. *In* Stuckey, P. J., éditeur : *CP*, volume 5202 de *Lecture Notes in Computer Science*, pages 463–477. Springer. 161

[Bessiere *et al.*, 2014] Bessiere, C., Coletta, R., Daoudi, A., Lazaar, N., Mechqrane, Y. et Bouyakhf, E. (2014). Boosting constraint acquisition via generalization queries. *In ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 99–104. 154

[Bessière et Cordier, 1993] Bessière, C. et Cordier, M.-O. (1993). Arc-consistency and arc-consistency again. *In* Fikes, R. et Lehnert, W. G., éditeurs : *AAAI*, pages 108–113. AAAI Press / The MIT Press. 35

[Bessière et Régin, 1996] Bessière, C. et Régin, J.-C. (1996). MAC and combined heuristics : Two reasons to forsake FC (and CBJ ?) on hard problems. *In* Freuder, E. C., éditeur : *CP*, volume 1118 de *Lecture Notes in Computer Science*, pages 61–75. Springer. 36, 38

[Bessière et Régin, 2001] Bessière, C. et Régin, J.-C. (2001). Refining the basic constraint propagation algorithm. *In* Nebel, B., éditeur : *IJCAI*, pages 309–315. Morgan Kaufmann. 36

[Bistarelli et Bonchi, 2007] Bistarelli, S. et Bonchi, F. (2007). Soft constraint based pattern mining. *Data Knowl. Eng.*, 62(1):118–137. 4, 71

[Bistarelli *et al.*, 1999] Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G. et Fargier, H. (1999). Semiring-based csps and valued csps : Frameworks, properties, and comparison. *Constraints*, 4(3):199–240. 39

[Böhm *et al.*, 2005] Böhm, K., Jensen, C. S., Haas, L. M., Kersten, M. L., Larson, P.-Å. et Ooi, B. C., éditeurs (2005). *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. ACM. 169, 171

[Bonchi et Lucchese, 2007] Bonchi, F. et Lucchese, C. (2007). Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60(2):377–399. 17

[Borgelt, 2003] Borgelt, C. (2003). Efficient implementations of apriori and eclat. *In ICDM Workshop FIMI*. 145

[Borning *et al.*, 1992] Borning, A., Freeman-Benson, B. N. et Wilson, M. (1992). Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270. 39

[Börzsönyi *et al.*, 2001] Börzsönyi, S., Kossmann, D. et Stocker, K. (2001). The skyline operator. *In* Georgakopoulos, D. et Buchmann, A., éditeurs : *ICDE*, pages 421–430. IEEE Computer Society. 4, 53, 54

[Boulicaut *et al.*, 2000] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2000). Approximation of frequency queries by means of free-sets. *In* Zighed, D. A., Komorowski, H. J. et Zytkow, J. M., éditeurs : *PKDD*, volume 1910 de *Lecture Notes in Computer Science*, pages 75–85. Springer. 19, 20

[Boulicaut *et al.*, 2003] Boulicaut, J.-F., Bykowski, A. et Rigotti, C. (2003). Free-sets : A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22. 20, 137

[Bringmann et Zimmermann, 2007] Bringmann, B. et Zimmermann, A. (2007). The chosen few : On identifying valuable patterns. *In ICDM*, pages 63–72. IEEE Computer Society. 26

[Burdick *et al.*, 2005] Burdick, D., Calimlim, M., Flannick, J., Gehrke, J. et Yiu, T. (2005). Mafia : A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504. 20

[Calders et Goethals, 2007] Calders, T. et Goethals, B. (2007). Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1):171–206. 20

[Caspard *et al.*, 2012] Caspard, N., Leclerc, B. et Monjardet, B. (2012). *Finite Ordered Sets Concepts, Results and Uses*. Cambridge University Press. Series Encyclopedia of Mathematics and Its Applications (No. 144). 144

[Cerf, 2010] Cerf, L. (2010). *Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations*. Thèse de doctorat, INSA-Lyon. 18, 31

[Chan *et al.*, 2006] Chan, C. Y., Jagadish, H. V., Tan, K., Tung, A. K. H. et Zhang, Z. (2006). Finding k-dominant skylines in high dimensional space. *In* Chaudhuri, S., Hristidis, V. et Polyzotis, N., éditeurs : *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 503–514. ACM. 154

[Cheung et Fu, 2004] Cheung, Y.-L. et Fu, A. W.-C. (2004). Mining frequent itemsets without support threshold : With and without item constraints. *IEEE Trans. Knowl. Data Eng.*, 16(9):1052–1069. 139

[Chomicki *et al.*, 2003] Chomicki, J., Godfrey, P., Gryz, J. et Liang, D. (2003). Skyline with presorting. *In ICDE*, pages 717–719. 54

[Chomicki *et al.*, 2005] Chomicki, J., Godfrey, P., Gryz, J. et Liang, D. (2005). Skyline with presorting : Theory and optimizations. *In Intelligent Information Systems*, pages 595–604. 53, 54

[Coquery *et al.*, 2012] Coquery, E., Jabbour, S., Saïs, L. et Salhi, Y. (2012). A sat-based approach for discovering frequent, closed and maximal patterns in a sequence. *In* Raedt, L. D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F. et Lucas, P. J. F., éditeurs : *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 de *Frontiers in Artificial Intelligence and Applications*, pages 258–263. IOS Press. 153

[Crémilleux et Boulicaut, 2002] Crémilleux, B. et Boulicaut, J.-F. (2002). Simplest rules characterizing classes generated by delta-free sets. *In 22nd Int. Conf. on Knowledge Based Systems and Applied Artificial Intelligence (ES'02)*, pages 33–46, Cambridge, UK. Springer. 140

[Crémilleux et Soulet, 2008] Crémilleux, B. et Soulet, A. (2008). Discovering knowledge from local patterns with global constraints. *In* Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y. et Gavrilova, M. L., éditeurs : *ICCSA (2)*, volume 5073 de *Lecture Notes in Computer Science*, pages 1242–1257. Springer. 27, 28, 30, 31, 138, 144, 161

[Dao *et al.*, 2013] Dao, T., Duong, K. et Vrain, C. (2013). A declarative framework for constrained clustering. *In* Blockeel, H., Kersting, K., Nijssen, S. et Zelezný, F., éditeurs : *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, volume 8190 de *Lecture Notes in Computer Science*, pages 419–434. Springer. 153

[De Raedt *et al.*, 2008] De Raedt, L., Guns, T. et Nijssen, S. (2008). Constraint programming for itemset mining. *In* Li, Y., Liu, B. et Sarawagi, S., éditeurs : *KDD*, pages 204–212. ACM. 3, 32, 47, 155

[De Raedt et Zimmermann, 2007] De Raedt, L. et Zimmermann, A. (2007). Constraint-based pattern set mining. *In SDM*, pages 237–248. SIAM. 25, 26, 31

[Debruyne, 1996] Debruyne, R. (1996). Arc-consistency in dynamic csps is no more prohibitive. *In ICTAI*, pages 299–307. 43

[Dong et Li, 1999] Dong, G. et Li, J. (1999). Efficient mining of emerging patterns : Discovering trends and differences. *In* Fayyad, U. M., Chaudhuri, S. et Madigan, D., éditeurs : *KDD*, pages 43–52. ACM. 15, 18

[Ehrgott *et al.*, 2009] Ehrgott, M., Fonseca, C. M., Gandibleux, X., Hao, J. et Sevaux, M., éditeurs (2009). *Evolutionary Multi-Criterion Optimization, 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings*, volume 5467 de *Lecture Notes in Computer Science*. Springer. 154

[Fan et Ramamohanarao, 2002] Fan, H. et Ramamohanarao, K. (2002). An efficient single-scan algorithm for mining essential jumping emerging patterns for classification. *In* Cheng, M.-S., Yu, P. S. et Liu, B., éditeurs : *PAKDD*, volume 2336 de *Lecture Notes in Computer Science*, pages 456–462. Springer. 140

[Fisher, 1987] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172. 23

[Freuder et Wallace, 1992] Freuder, E. C. et Wallace, R. J. (1992). Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70. 39

[Fu *et al.*, 2000] Fu, A. W.-C., w. Kwong, R. W. et Tang, J. (2000). Mining n-most interesting itemsets. *In* Ras, Z. W. et Ohsuga, S., éditeurs : *ISMIS*, volume 1932 de *Lecture Notes in Computer Science*, pages 59–67. Springer. 26, 27

[Garofalakis *et al.*, 1999] Garofalakis, M. N., Rastogi, R. et Shim, K. (1999). Spirit : Sequential pattern mining with regular expression constraints. *In* Atkinson, M. P., Orlowska, M. E., Valduriez, P., Zdonik, S. B. et Brodie, M. L., éditeurs : *VLDB*, pages 223–234. Morgan Kaufmann. 71

[Gavanelli, 2002] Gavanelli, M. (2002). An algorithm for multi-criteria optimization in csps. *In* van Harmelen, F., éditeur : *ECAI*, pages 136–140. IOS Press. 55

[Gecode Team, 2006] Gecode Team (2006). Gecode : Generic constraint development environment. Available from `http://www.gecode.org`. 159

[Geerts *et al.*, 2004] Geerts, F., Goethals, B. et Mielikäinen, T. (2004). Tiling databases. *In* [Suzuki et Arikawa, 2004], pages 278–289. 15, 23, 26, 31, 138, 140

[Geng et Hamilton, 2006] Geng, L. et Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Comput. Surv.*, 38(3). 13, 14

[Giacometti *et al.*, 2011] Giacometti, A., Marcel, P. et Soulet, A. (2011). A relational view of pattern discovery. *In* Yu, J. X., Kim, M.-H. et Unland, R., éditeurs : *DASFAA (1)*, volume 6587 de *Lecture Notes in Computer Science*, pages 153–167. Springer. 27

[Giacometti *et al.*, 2009] Giacometti, A., Miyaneh, E. K., Marcel, P. et Soulet, A. (2009). A framework for pattern-based global models. *In* Corchado, E. et Yin, H., éditeurs : *IDEAL*, volume 5788 de *Lecture Notes in Computer Science*, pages 433–440. Springer. 27

[Ginsberg, 1993] Ginsberg, M. L. (1993). Dynamic backtracking. *J. Artif. Intell. Res. (JAIR)*, 1:25–46. 43

[Golomb et Baumert, 1965] Golomb, S. W. et Baumert, L. D. (1965). Backtrack programming. *J. ACM*, 12(4):516–524. 36

[Gouda et Zaki, 2005] Gouda, K. et Zaki, M. J. (2005). Genmax : An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3):223–242. 20

[Gray *et al.*, 1997] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. et Pirahesh, H. (1997). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53. 56

[Gunopulos *et al.*, 1997] Gunopulos, D., Khardon, R., Mannila, H. et Toivonen, H. (1997). Data mining, hypergraph transversals, and machine learning. *In* Mendelzon, A. O. et Özsoyoglu, Z. M., éditeurs : *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*, pages 209–216. ACM Press. 137

[Guns, 2012] Guns, T. (2012). *Declarative Pattern Mining using Constraint Programming.* Thèse de doctorat, University of Leuven - KU Leuven. 32

[Guns *et al.*, 2013a] Guns, T., Dries, A., Tack, G., Nijssen, S. et Raedt, L. D. (2013a). Miningzinc : A modeling language for constraint-based mining. *In* Rossi, F., éditeur : *IJCAI*. IJCAI/AAAI. 3

[Guns *et al.*, 2011] Guns, T., Nijssen, S. et De Raedt, L. (2011). Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983. 3

[Guns *et al.*, 2013b] Guns, T., Nijssen, S. et De Raedt, L. (2013b). k-pattern set mining under constraints. *IEEE Trans. Knowl. Data Eng.*, 25(2):402–418. 3, 22, 25, 31

[Han *et al.*, 2002] Han, J., Wang, J., Lu, Y. et Tzvetkov, P. (2002). Mining top-k frequent closed patterns without minimum support. *In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 211–218. IEEE Computer Society. 26, 27, 138

[Hand, 2002] Hand, D. J. (2002). Pattern detection and discovery. *In* Hand, D. J., Adams, N. M. et Bolton, R. J., éditeurs : *Pattern Detection and Discovery*, volume 2447 de *Lecture Notes in Computer Science*, pages 1–12. Springer. 13

[Hansen *et al.*, 2009] Hansen, K., Mika, S., Schroeter, T., Sutter, A., ter Laak, A., Steger-Hartmann, T., Heinrich, N. et Müller, K.-R. (2009). Benchmark data set for in silico prediction of ames mutagenicity. *Journal of Chemical Information and Modeling*, 49(9):2077–2081. 125

[Haralick et Elliott, 1980] Haralick, R. M. et Elliott, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artif. Intell.*, 14(3):263–313. 38

[Jabbour *et al.*, 2013a] Jabbour, S., Sais, L. et Salhi, Y. (2013a). Boolean satisfiability for sequence mining. *In* He, Q., Iyengar, A., Nejdl, W., Pei, J. et Rastogi, R., éditeurs : *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 649–658. ACM. 153

[Jabbour *et al.*, 2013b] Jabbour, S., Sais, L. et Salhi, Y. (2013b). The top-k frequent closed itemset mining using top-k SAT problem. *In Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, pages 403–418. 138

[Jin *et al.*, 2004] Jin, W., Han, J. et Ester, M. (2004). Mining thick skylines over large databases. *In* Boulicaut, J.-F., Esposito, F., Giannotti, F. et Pedreschi, D., éditeurs : *PKDD*, volume 3202 de *Lecture Notes in Computer Science*, pages 255–266. Springer. 55

[Jussien *et al.*, 2000] Jussien, N., Debruyne, R. et Boizumault, P. (2000). Maintaining arc-consistency within dynamic backtracking. *In Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings*, pages 249–261. 43

[Ke *et al.*, 2009] Ke, Y., Cheng, J. et Yu, J. X. (2009). Top-k correlative graph mining. *In SDM*, pages 1038–1049. SIAM. 68

[Kearns et Vazirani, 1994] Kearns, M. J. et Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA. 23

[Kemmar *et al.*, 2014] Kemmar, A., Ugarte, W., Loudni, S., Charnois, T., Lebbah, Y., Boizumault, P. et Crémilleux, B. (2014). Mining relevant sequence patterns with cp-based framework. *In 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'2014), 10-12 November 2014, Limassol, Cyprus*. 153

[Khiari, 2012] Khiari, M. (2012). *Découverte de motifs n-aires utilisant la programmation par contraintes*. Thèse de doctorat, GREYC - Université de Caen Basse-Normandie. 22, 24, 28, 32, 46

[Khiari *et al.*, 2009] Khiari, M., Boizumault, P. et Crémilleux, B. (2009). Local constraint-based mining and set constraint programming for pattern discovery. *In From Local Patterns to Global Models (LeGo-09), ECML/PKDD-09 Workshop*, pages 61–76, Bled, Slovenia. 3

[Khiari *et al.*, 2010a] Khiari, M., Boizumault, P. et Crémilleux, B. (2010a). Allier csps et motifs locaux pour la découverte de motifs sous contraintes n-aires. *In* Yahia, S. B. et Petit, J.-M., éditeurs : *EGC*, volume RNTI-E-19 de *Revue des Nouvelles Technologies de l'Information*, pages 199–210. Cépaduès-éditions. 3

[Khiari *et al.*, 2010b] Khiari, M., Boizumault, P. et Crémilleux, B. (2010b). Constraint programming for mining n-ary patterns. *In* Cohen, D., éditeur : *CP*, volume 6308 de *Lecture Notes in Computer Science*, pages 552–567. Springer. 3, 22, 23, 25, 31, 32, 48, 77, 78

[Khiari *et al.*, 2012] Khiari, M., Lallouet, A. et Vautard, J. (2012). Extraction de motifs sous contraintes quantifiées. *In 8-èmes Journées Francophones de Programmation par Contraintes (JFPC 2012)*, Toulouse, France. 28, 147

[Knobbe *et al.*, 2008] Knobbe, A., Crémilleux, B., Fürnkranz, J. et Scholz, M. (2008). From local patterns to global models : The lego approach to data mining. *In Int. Workshop LeGo co-located with ECML/PKDD'08*, pages 1–16, Antwerp, Belgium. 27

[Knobbe et Ho, 2006a] Knobbe, A. J. et Ho, E. K. Y. (2006a). Maximally informative k-itemsets and their efficient discovery. *In* Eliassi-Rad, T., Ungar, L. H., Craven, M. et Gunopulos, D., éditeurs : *KDD*, pages 237–244. ACM. 139

[Knobbe et Ho, 2006b] Knobbe, A. J. et Ho, E. K. Y. (2006b). Pattern teams. *In* Fürnkranz, J., Scheffer, T. et Spiliopoulou, M., éditeurs : *PKDD*, volume 4213 de *Lecture Notes in Computer Science*, pages 577–584. Springer. 26

[Kossmann *et al.*, 2002] Kossmann, D., Ramsak, F. et Rost, S. (2002). Shooting stars in the sky : An online algorithm for skyline queries. *In VLDB*, pages 275–286. 54

[Kung *et al.*, 1975] Kung, H. T., Luccio, F. et Preparata, F. P. (1975). On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476. 53

[Lam *et al.*, 2014] Lam, H. T., Pei, W., Prado, A., Jeudy, B. et Fromont, É. (2014). Mining top-k largest tiles in a data stream. *In Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 82–97. 154

[Le et Silva, 2007] Le, K. et Silva, D. L. (2007). Obtaining better non-dominated sets using volume dominance. *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore*, pages 3119–3126. IEEE. 154

[Li *et al.*, 2005] Li, J., Fu, A. W.-C., He, H., Chen, J., Jin, H., McAullay, D., Williams, G. J., Sparks, R. et Kelman, C. (2005). Mining risk patterns in medical data. *In* Grossman, R., Bayardo, R. J. et Bennett, K. P., éditeurs : *KDD*, pages 770–775. ACM. 139

[Li *et al.*, 2007] Li, J., Liu, G. et Wong, L. (2007). Mining statistically important equivalence classes and delta-discriminative emerging patterns. *In* Berkhin, P., Caruana, R. et Wu, X., éditeurs : *KDD*, pages 430–439. ACM. 20

[Mackworth, 1977] Mackworth, A. K. (1977). Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118. 35

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *In* Le Cam, L. M. et Neyman, J., éditeurs : *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, pages 281–297. University of California Press, Berkeley, CA, USA. 103

[Mannila et Toivonen, 1996] Mannila, H. et Toivonen, H. (1996). Multiple uses of frequent sets and condensed representations (extended abstract). *In* Simoudis, E., Han, J. et Fayyad, U. M., éditeurs : *KDD*, pages 189–194. AAAI Press. 19

[Mannila et Toivonen, 1997] Mannila, H. et Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258. 14, 17, 32, 137

[Matousek, 1991] Matousek, J. (1991). Computing dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278. 53

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artif. Intell.*, 18(2):203–226. 17

[Mohr et Henderson, 1986] Mohr, R. et Henderson, T. C. (1986). Arc and path consistency revisited. *Artif. Intell.*, 28(2):225–233. 35

[Morik *et al.*, 2005] Morik, K., Boulicaut, J.-F. et Siebes, A., éditeurs (2005). *Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers*, volume 3539 de *Lecture Notes in Computer Science*. Springer. 13, 31

[Métivier, 2010] Métivier, J.-P. (2010). *Relaxation de contraintes globales : Mise en œuvre et Application*. Thèse de doctorat, GREYC - Université de Caen Basse-Normandie. 42

[Métivier *et al.*, 2011] Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M. et Loudni, S. (2011). A constraint-based language for declarative pattern discovery. *In* Spiliopoulou, M., Wang, H., Cook, D. J., Pei, J., Wang, W., Zaïane, O. R. et Wu, X., éditeurs : *ICDM Workshops*, pages 1112–1119. IEEE. 3

[Métivier *et al.*, 2012] Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M. et Loudni, S. (2012). A constraint language for declarative pattern discovery. *In* Ossowski, S. et Lecca, P., éditeurs : *SAC*, pages 119–125. ACM. 3

[Ng *et al.*, 1998] Ng, R. T., Lakshmanan, L. V. S., Han, J. et Pang, A. (1998). Exploratory mining and pruning optimizations of constrained association rules. *In SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 13–24. 17

[Novak *et al.*, 2009] Novak, P. K., Lavrac, N. et Webb, G. I. (2009). Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403. 15, 139

[Négrevergne *et al.*, 2013] Négrevergne, B., Dries, A., Guns, T. et Nijssen, S. (2013). Dominance programming for itemset mining. *In* Xiong, H., Karypis, G., Thuraisingham, B. M., Cook, D. J. et Wu, X., éditeurs : *ICDM*, pages 557–566. IEEE. 144, 145

[Padmanabhan et Tuzhilin, 1998] Padmanabhan, B. et Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. *In* Agrawal, R., Stolorz, P. E. et Piatetsky-Shapiro, G., éditeurs : *KDD*, pages 94–100. AAAI Press. 23, 24, 25, 31

[Papadias *et al.*, 2008] Papadias, D., Yiu, M. L., Mamoulis, N. et Tao, Y. (2008). Nearest neighbor queries in network databases. *In* Shekhar, S. et Xiong, H., éditeurs : *Encyclopedia of GIS*, pages 772–776. Springer. 53

[Parida et Ramakrishnan, 2005] Parida, L. et Ramakrishnan, N. (2005). Redescription mining : Structure theory and algorithms. *In* Veloso, M. M. et Kambhampati, S., éditeurs : *AAAI*, pages 837–844. AAAI Press / The MIT Press. 23

[Pasquier *et al.*, 1999a] Pasquier, N., Bastide, Y., Taouil, R. et Lakhal, L. (1999a). Discovering frequent closed itemsets for association rules. *In* Beeri, C. et Buneman, P., éditeurs : *ICDT*, volume 1540 de *Lecture Notes in Computer Science*, pages 398–416. Springer. 19, 20, 137

[Pasquier *et al.*, 1999b] Pasquier, N., Bastide, Y., Taouil, R. et Lakhal, L. (1999b). Efficient mining of association rules using closed itemset lattices. *Inf. Syst.*, 24(1):25–46. 20

[Pei *et al.*, 2007] Pei, J., Fu, A. W.-C., Lin, X. et Wang, H. (2007). Computing compressed multidimensional skyline cubes efficiently. *In* Chirkova, R., Dogac, A., Özsu, M. T. et Sellis, T. K., éditeurs : *ICDE*, pages 96–105. IEEE. 56

[Pei *et al.*, 2005] Pei, J., Jin, W., Ester, M. et Tao, Y. (2005). Catching the best views of skyline : A semantic approach based on decisive subspaces. *In* [Böhm *et al.*, 2005], pages 253–264. 55

[Pei *et al.*, 2006] Pei, J., Yuan, Y., Lin, X., Jin, W., Ester, M., Liu, Q., Wang, W., Tao, Y., Yu, J. X. et Zhang, Q. (2006). Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4):1335–1381. 55

[Pennerath et Napoli, 2009] Pennerath, F. et Napoli, A. (2009). The model of most informative patterns and its application to knowledge extraction from graph databases. *In* Buntine, W. L., Grobelnik, M., Mladenic, D. et Shawe-Taylor, J., éditeurs : *ECML/PKDD (2)*, volume 5782 de *Lecture Notes in Computer Science*, pages 205–220. Springer. 4

[Petit, 2002] Petit, T. (2002). *Modélisation et Algorithmes de Résolution de Problèmes Sur-Contraints.* Thèse de doctorat, LIRMM - Université de Montpellier II. 39, 40, 41, 60

[Petit *et al.*, 2000] Petit, T., Régin, J.-C. et Bessière, C. (2000). Meta-constraints on violations for over constrained problems. *In ICTAI*, pages 358–365. IEEE Computer Society. 40

[Poezevara *et al.*, 2010] Poezevara, G., Cuissart, B., Crémilleux, B., Lozano, S., Halm-Lemeille, M.-P., Lepailleur, E. L.-F. A., Bisell-Siders, R., Rault, S. et Bureau, R. (2010). Introduction of jumping fragments in combination with qsars for the assessment of classification in ecotoxicology. *Journal of Chemical Information and Modeling (JCIM)*, pages 1330–1339. 15, 72

[Poezevara *et al.*, 2011] Poezevara, G., Cuissart, B. et Crémilleux, B. (2011). Extracting and summarizing the frequent emerging graph patterns from a dataset of graphs. *J. Intell. Inf. Syst.*, 37(3):333–353. 72

[Raïssi *et al.*, 2010] Raïssi, C., Pei, J. et Kister, T. (2010). Computing closed skycubes. *PVLDB*, 3(1):838–847. 56

[Régin *et al.*, 2000] Régin, J.-C., Petit, T., Bessière, C. et Puget, J.-F. (2000). An original constraint based approach for solving over constrained problems. *In* Dechter, R., éditeur : *CP*, volume 1894 de *Lecture Notes in Computer Science*, pages 543–548. Springer. 60

[Sabin et Freuder, 1994] Sabin, D. et Freuder, E. C. (1994). Contradicting conventional wisdom in constraint satisfaction. *In* Borning, A., éditeur : *PPCP*, volume 874 de *Lecture Notes in Computer Science*, pages 10–20. Springer. 36, 38

[Schiex *et al.*, 1995] Schiex, T., Fargier, H. et Verfaillie, G. (1995). Valued constraint satisfaction problems : Hard and easy problems. *In IJCAI (1)*, pages 631–639. Morgan Kaufmann. 39

[Soulet, 2006] Soulet, A. (2006). *Un cadre générique de découverte de motifs sous contraintes basées sur des primitives*. Thèse de doctorat, GREYC - Université de Caen Basse-Normandie. 20, 22, 27

[Soulet et Crémilleux, 2005] Soulet, A. et Crémilleux, B. (2005). An efficient framework for mining flexible constraints. *In* Ho, T. B., Cheung, D. W.-L. et Liu, H., éditeurs : *PAKDD*, volume 3518 de *Lecture Notes in Computer Science*, pages 661–671. Springer. 18, 31

[Soulet et Crémilleux, 2005] Soulet, A. et Crémilleux, B. (2005). Optimizing constraint-based mining by automatically relaxing constraints. *In ICDM*, pages 777–780. IEEE Computer Society. 71

[Soulet et Crémilleux, 2008] Soulet, A. et Crémilleux, B. (2008). Adequate condensed representations of patterns. *Data Min. Knowl. Discov.*, 17(1):94–110. 20

[Soulet *et al.*, 2004] Soulet, A., Crémilleux, B. et Rioult, F. (2004). Condensed representation of emerging patterns. *In* Dai, H., Srikant, R. et Zhang, C., éditeurs : *PAKDD*, volume 3056 de *Lecture Notes in Computer Science*, pages 127–132. Springer. 20, 140

[Soulet *et al.*, 2011] Soulet, A., Raïssi, C., Plantevit, M. et Crémilleux, B. (2011). Mining dominant patterns in the sky. *In* Cook, D. J., Pei, J., Wang, W., Zaïane, O. R. et Wu, X., éditeurs : *ICDM*, pages 655–664. IEEE. 22, 85, 86, 88, 91, 92, 96, 137

[Steuer, 1992] Steuer, R. E. (1992). *Multiple Criteria Optimization : Theory, Computation and Application*. Radio e Svyaz, Moscow, 504 pp. (in Russian). 53

[Suzuki, 2002] Suzuki, E. (2002). Undirected discovery of interesting exception rules. *IJPRAI*, 16(8): 1065–1086. 23, 24, 25, 31

[Suzuki et Arikawa, 2004] Suzuki, E. et Arikawa, S., éditeurs (2004). *Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings*, volume 3245 de *Lecture Notes in Computer Science*. Springer. 166, 171

[Szathmary *et al.*, 2007] Szathmary, L., Napoli, A. et Valtchev, P. (2007). Towards rare itemset mining. *In ICTAI (1)*, pages 305–312. IEEE Computer Society. 77

[Tan *et al.*, 2001] Tan, K.-L., Eng, P.-K. et Ooi, B. C. (2001). Efficient progressive skyline computation. *In* Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K. et Snodgrass, R. T., éditeurs : *VLDB*, pages 301–310. Morgan Kaufmann. 53, 54

[Tao *et al.*, 2007] Tao, Y., Xiao, X. et Pei, J. (2007). Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. Knowl. Data Eng.*, 19(8):1072–1088. 154

[Ugarte *et al.*, 2014a] Ugarte, W., Boizumault, P., Crémilleux, B. et Loudni, S. (2014a). Modélisation et extraction de motifs optimaux. *In 19ème congrès national sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'2014), 30 juin-4 juillet 2014, Rouen, France*. 6

[Ugarte *et al.*, 2012a] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2012a). Soft threshold constraints for pattern mining. *In 15th International Conference on Discovery Science, (DS'2012), 29-31 October, 2012, Lyon, France*, volume 7569, pages 313–327. Springer. 5, 138

[Ugarte *et al.*, 2014b] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2014b). Computing skypattern cubes. *In 21st European Conference on Artificial Intelligence (ECAI'2014), 18-22 August 2014, Prague, Czech Republic*, volume 263 de *Frontiers in Artificial Intelligence and Applications*, pages 903–908. IOS Press. 6

[Ugarte *et al.*, 2014c] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2014c). Computing skypattern cubes using relaxation. *In 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'2014), 10-12 November 2014, Limassol, Cyprus*. 6

[Ugarte *et al.*, 2013a] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2013a). Découverte des soft-skypatterns avec une approche PPC. *In 13e Conférence Francophone sur l'Extraction et la Gestion des Connaissances (EGC'2013), 29 janvier - 01 février 2013, Toulouse, France*, volume RNTI-E-24 de *Revue des Nouvelles Technologies de l'Information*, pages 217–228. Hermann-Éditions. 6

[Ugarte *et al.*, 2013b] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2013b). Soft constraints for pattern mining. *Journal of Intelligent Information Systems*, pages 1–29. `http://link.springer.com/article/10.1007/s10844-013-0281-4` [Online]. 5, 138

[Ugarte *et al.*, 2014d] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2014d). Mining (soft-) skypatterns using dynamic CSP. *In 11th International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming (CP) (CPAIOR'2014), 19-23 May, 2014, Cork, Ireland*, pages 71–87. 6, 137

[Ugarte *et al.*, 2012b] Ugarte, W., Boizumault, P., Loudni, S. et Crémilleux, B. (2012b). Extraction de motifs sous contraintes souples de seuil. *In 8èmes Journées Francophones de Programmation par Contraintes (JFPC'2012), 22-24 Mai 2012, Toulouse, France*, pages 302–311. 5

[Ugarte *et al.*, 2013c] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B. et Lepailleur, A. (2013c). Mining (Soft-) Skypatterns using Constraint Programming. *In ECML/PKDD workshop on Languages for Data Mining and Machine Learning (LML'2013),23 September 2013, Prague, Czech Republic*, pages 20–34, Prague, Czech Republic. 6

[Uno *et al.*, 2004] Uno, T., Asai, T., Uchida, Y. et Arimura, H. (2004). An efficient algorithm for enumerating closed patterns in transaction databases. *In* [Suzuki et Arikawa, 2004], pages 16–31. 20, 145

[Verfaillie et Jussien, 2005] Verfaillie, G. et Jussien, N. (2005). Constraint solving in uncertain and dynamic environments : A survey. *Constraints*, 10(3):253–281. 42

[Wang *et al.*, 2005] Wang, J., Han, J., Lu, Y. et Tzvetkov, P. (2005). Tfp : An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Trans. Knowl. Data Eng.*, 17(5):652–664. 68

[Wang *et al.*, 2004] Wang, Z., Fan, H. et Ramamohanarao, K. (2004). Exploiting maximal emerging patterns for classification. *In* Webb, G. I. et Yu, X., éditeurs : *Australian Conference on Artificial Intelligence*, volume 3339 de *Lecture Notes in Computer Science*, pages 1062–1068. Springer. 140

[Xia *et al.*, 2008] Xia, T., Zhang, D. et Tao, Y. (2008). On skylining with flexible dominance relation. *In* Alonso, G., Blakeley, J. A. et Chen, A. L. P., éditeurs : *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 1397–1399. IEEE. 154

[Xin *et al.*, 2005] Xin, D., Han, J., Yan, X. et Cheng, H. (2005). Mining compressed frequent-pattern sets. *In* [Böhm *et al.*, 2005], pages 709–720. 139

[Yiu et Mamoulis, 2009] Yiu, M. L. et Mamoulis, N. (2009). Multi-dimensional top-k dominating queries. *VLDB J.*, 18(3):695–718. 154

[Yuan *et al.*, 2005] Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J. X. et Zhang, Q. (2005). Efficient computation of the skyline cube. *In* [Böhm *et al.*, 2005], pages 241–252. 55, 56

[Zaki et Hsiao, 2002] Zaki, M. J. et Hsiao, C.-J. (2002). Charm : An efficient algorithm for closed itemset mining. *In* Grossman, R. L., Han, J., Kumar, V., Mannila, H. et Motwani, R., éditeurs : *SDM*, pages 457–473. SIAM. 20

[Zimmermann, 2009] Zimmermann, A. (2009). *Mining Sets of Patterns.* Thèse de doctorat, University Freiburg, Germany. 22, 25

Mots-clés indexation RAMEAU : Exploration de données, Programmation par contraintes, Contraintes (intelligence artificielle), Bases de données.

Discipline : Informatique et applications.

Laboratoire : GREYC CNRS UMR 6072, Campus Côte de Nacre, Boulevard du Maréchal Juin BP 5186 - 14032 Caen CEDEX.